Certificação Linux

Guia prático para a prova LPIC-1 101

Casa do Código | alura

JULIANO RAMOS

Sumário

- ISBN
- Apresentação
- Sobre o livro
- 1. Introdução
- 2. Comandos GNU e UNIX
- 3. Arquivos de texto
- 4. Gerenciamento de arquivos
- 5. Redirecionamento de entrada e saída
- 6. Gerenciamento de processos
- 7. Pesquisar arquivos de texto com expressões regulares
- 8. Editor de texto VIM
- 9. Arquitetura do sistema
- 10. O boot do sistema
- 11. Particionamento de discos
- 12. Configuração de boot loader
- 13. Gerenciamento de pacotes
- 14. Dispositivos, sistemas de arquivos linux e FHS
- 15. Agradecimentos

ISBN

Impresso: 978-85-5519-292-0

Digital: 978-85-5519-295-1

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

Apresentação

O conhecimento sobre Linux é imprescindível para várias oportunidades de trabalho na área de computação, incluindo Analistas de Infraestrutura, Analistas de Segurança, Peritos Forense Computacionais, Engenheiros de Rede, Analistas de Computação em Nuvem, Engenheiros DevOps e SRE. E não apenas oportunidades na área de Infraestrutura, muitos desenvolvedores vão se aproveitar de várias ferramentas que um desktop Linux pode oferecer e, mais ainda, vão precisar compreender bem o Linux para trabalhar com contêineres e virtualização, dentre várias possibilidades. Mas quero falar nesse momento como profissional da área de Infraestrutura, área com a qual lido desde a década de 90 e na qual sou muito realizado.

Nós, da área de Infraestrutura, somos muitas vezes preparados para o “aguardar pelo pior”. Muitas vezes a maior parte do trabalho é “apagar um incêndio e evitar que o fogo se espalhe”. Eu mesmo sempre disse a meus alunos de gerência de redes que o trabalho do Gerente é “evitar que o telefone toque”, porque ninguém liga quando tudo dá certo. Eu fui preparado assim para o início da minha carreira.

Mas tive a oportunidade de trabalhar com o Professor Juliano a quem admiro muito e que me apresentou o outro lado da moeda. Estar preparado para enfrentar um problema quando ele aparece é importante, mas pensar positivo e ter uma atitude sempre preditiva é algo tão útil quanto. E muito disso é a base para atuar como Engenheiro DevOps, cargo que assumo como um presente que me foi dado e que tento honrar mais e mais todos os dias.

Durante algum tempo, o Professor Juliano enfrentou problemas que fariam qualquer um desistir, mas a frase “a vida é muito boa e sempre vai dar certo” era tão frequente que muitos diziam ser o seu mantra. Na realidade, não consigo mais pensar nessa frase sem associá-la imediatamente aos vários vídeos instrutivos a que tive

acesso acompanhando seu bem-sucedido canal. Conheci o Professor Juliano como um de seus alunos, trabalhamos juntos em algum momento e hoje posso chamá-lo de amigo.

Mas não é por ser amigo dele que estou escrevendo essa apresentação. Na realidade, negaria a muitos dos meus amigos esse ato se não tivesse plena convicção de que estamos diante de um conteúdo excelente. E digo isso por muitas razões: por ter sido aluno nos cursos do Professor Juliano, por ter trabalhado com ele e ter sido uma das testemunhas de seu empenho e compromisso profissional com todas as responsabilidades que assume e por ter sido eu mesmo um dos revisores do livro. Como se não bastassem as suas próprias certificações, que servem justamente para isso: certificar seu domínio no assunto.

Por tudo isso, acredito que todos que desejam aprofundar seus conhecimentos em Linux para melhorar seu desempenho no trabalho, para buscar uma nova posição no mercado de trabalho ou para tirar a certificação vão encontrar no livro *Certificação Linux: Guia prático para a prova LPIC-1 101* um ponto de apoio certo e confiável.

Caríssimos leitores e leitoras, que são a razão de todo o trabalho aqui dispensado, tenho algumas poucas recomendações que podem ser úteis durante a sua leitura. Para os vários comandos que você vai aprender neste livro, é importantíssimo que os pratique. Faça pequenos exemplos para compreender a utilidade de cada comando e atribua para si mesmo alguns pequenos exercícios para evoluir seu conhecimento.

Sempre que possível combine os comandos de modo que dois comandos juntos possam resolver um problema que cada um, sozinho, não seria capaz. Tente observar os cenários em que cada comando é melhor aplicado. Aproveite a oportunidade para evoluir seus conhecimentos em Shell Script e veja como é contagiante. Inclusive, tentar entender como alguém usa um comando em um

Script é outro desafio que traz muitos e gratos aprendizados. Recomendo que procure no GitHub por arquivos .sh para isso.

Em alguns momentos, um comando vai se mostrar simples e dispensável. É natural que em alguns casos se pense assim, mas logo você vai observar que cada comando é como uma engrenagem, que, por menor e frágil que pareça, tem sua utilidade específica. Pense também que muitos desses comandos foram criados na mesma época em que o próprio Linux foi criado (na década de 1980) ou mesmo já existiam. E muitos desses comandos convivem com outros que foram criados por todos esses anos para diferentes sistemas operacionais e cenários de utilização. Então não espere um comportamento sempre uniforme e saídas sempre parecidas ao comparar os comandos.

Por fim, junte-se a todos os profissionais que buscam aprimorar seus conhecimentos em Linux para aumentar sua produtividade e para se tornar um profissional melhor. E se após ler e estudar todos os conhecimentos ofertados neste livro você deseje evoluir mais, não deixe de procurar outros títulos do Professor Juliano que respira Linux e tem a bondade de compartilhar esse conhecimento conosco.

Cícero Woshington Saraiva Leite

Mestre em Engenharia de Software e Engenheiro DevOps, janeiro de 2022.

Sobre o livro

Este livro contém todo o conteúdo necessário para você ser aprovado no exame 101 da Certificação Linux LPIC-1 prova 5.0.

O material foi escrito como um guia rápido e direto para quem deseja realizar a prova e ainda não se sente seguro, ou gostaria de fixar seus estudos.

O livro é resultado dos meus mais de treze anos ministrando treinamento preparatório de linux LPIC e, a cada aluno, cada aluna, que conquista a certificação, obtenho feedback sobre o exame, entendendo as dificuldades e facilidades das provas. Os capítulos foram escritos em uma linguagem simples e direta, com foco 100% no exame.

O livro é para iniciantes em linux e para quem já trabalha com o sistema, mas deseja comprovar seu conhecimento com uma certificação internacional.

Capítulo 1
Introdução

O mercado da Tecnologia da Informação se diferencia das outras áreas porque está em constante e rápida evolução. Normalmente a cada seis meses uma nova versão de software supera a anterior. Por motivos como este, nem sempre um aluno que se gradua em cursos de tecnologia está apto para o mercado de trabalho.

Para preencher esta lacuna no mercado, grandes empresas como Microsoft, IBM, Novell, Red Hat, Cisco, Google, entre outras, se reuniram através de associações ou programas próprios de certificações, com o objetivo de capacitar profissionais para que atuem com eficácia nas suas plataformas.

Tratando-se de linux, a maior e mais reconhecida certificação é a LPI - Linux Professional Institute. Este livro tem como objetivo abordar o conteúdo da primeira prova da certificação LPIC-1, denominada de prova 101.

Sobre a LPI

A LPI possui quatro níveis de certificação. O primeiro é o *Linux Essentials*, que tem foco em usuários iniciantes. O segundo nível é chamado de *LPIC-1 - Linux Administrator*, o qual forma profissionais para atuarem como analistas e administradores de sistemas Linux. O próximo nível é chamado de *LPIC-2 - Linux Engineer*, que é a continuação direta da formação do Administrador Linux nível 1, e seu objetivo é formar administradores de redes e servidores linux. O último nível de certificação é chamado de *LPIC-3* e este é dividido em três áreas de atuação, sendo elas: Segurança, Complementação mista (*mixed*) e Virtualização. Você pode optar por qualquer uma destas áreas ou, se desejar, obter as três certificações do nível 3. Veja uma descrição das certificações disponíveis na LPI:

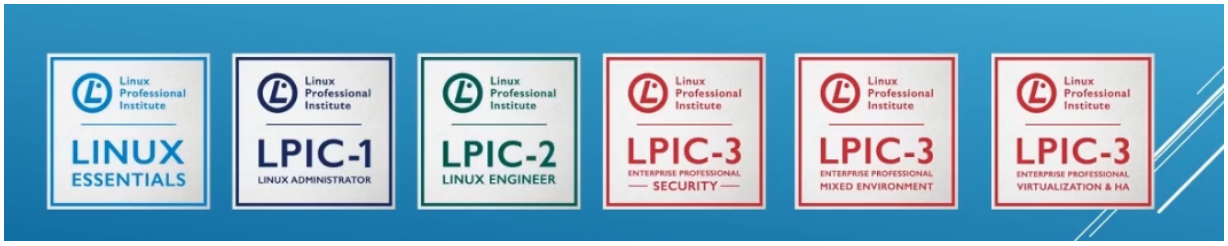

Figura 1.1: Certificações Linux.

Neste livro, vamos abordar a certificação LPIC-1, que tem como objetivo formar administradores de sistemas linux. A certificação Linux Essentials não é um pré-requisito para que você possa realizar as provas e ser um certificado nível 1. Como a prova Linux Essentials não forma profissionais, a LPIC-1 é a mais indicada para quem deseja entrar no mercado de trabalho. A LPIC-1 é um pré-requisito para que você possa obter a certificação nível 2 - LPIC-2, que por sua vez é um pré-requisito para que você possa obter o nível 3 - LPIC-3.

Duas provas para o nível 1

Para que você seja um certificado LPIC-1 (Linux Administrator) é necessário que você realize duas provas. As nomenclaturas destes exames são: LPIC 101-400 e LPIC 102-400. Cada exame possui 60 questões e tem uma hora e meia de duração. Cerca de 75% das questões são de múltipla escolha, sendo que existe somente uma opção correta. Algumas apresentarão um cenário onde alguma medida administrativa precisa ser tomada e, em outras, pergunta-se qual o comando apropriado para uma determinada tarefa. O exame testa seu conhecimento teórico, comandos e suas opções mais comuns. Ao contrário do que muitos propagam, não é uma prova difícil, porém, é necessário ter dedicação e preparação, como qualquer coisa na vida.

Posso realizar somente um exame?

Você pode realizar somente uma prova, porém não será certificado. A certificação só será entregue quando você receber aprovação nas

duas provas, sendo que o exame 102 tem como pré-requisito o exame 101.

Como marcar a prova?

Você pode marcar o seu exame em um centro Pearson Vue, que é uma corporação mundialmente famosa que aplica testes de exames de tecnologia ao redor do mundo, estando espalhados em escolas, empresas e universidades. O centro Person Vue é independente, o que significa que nenhuma escola ou universidade pode obrigá-lo a realizar seus cursos preparatórios para os exames de certificação. Você pode encontrar um centro Pearson Vue em sua cidade pelo site: http://www.pearsonvue.com/ Selecione: Tecnologia da Informação - Linux Professional Institute e, depois, Brasil.

Para comprar o voucher da prova, é preciso que você tenha um cadastro no site da LPI (http://www.lpi.org/), que é realizado de forma gratuita. Após esse cadastro, você terá um número de identificação, chamado de LPI-ID. No momento da aquisição de seu voucher para realizar as provas, este número será solicitado.

Como é o ambiente da prova?

As salas são fechadas e isoladas, sem distrações. Você não pode entrar com nenhum dispositivo eletrônico, agenda e nenhuma anotação. O exame é monitorado e pode ser cancelado em caso de alguma suspeita de "cola".

Dicas para nosso estudo

Como o exame cobra tanto o padrão de pacotes usado no Red Hat (RPM) e o padrão usado no Debian (DEB), é altamente recomendável que você possua duas distribuições que atendam estes pacotes disponíveis durante a leitura do livro. Recomendo que instale o Virtualbox, que é gratuito, e faça download e instalação das distribuições: Xubuntu/Debian e Centos/Red Hat.

A importância dos simulados

Fazer um simulado vai lhe dar uma ideia exata sobre o tipo de questão que você vai encontrar no exame LPI. Eu mantenho um simulado comentado no site http://www.certificacoes.net.br/, que, além de mostrar seus erros e acertos, fornece uma explicação sobre a questão. O simulado deve ser realizado quando você concluir este livro, que aborda o exame LPIC-1 101.

Capítulo 2
Comandos GNU e UNIX

Este livro está organizado segundo o programa de conteúdos oficiais para a certificação LPIC-1. Dessa forma, a pessoa candidata encontrará exatamente os temas que são abordados no exame de certificação, na profundidade que é exigida para a prova. A organização dos tópicos foi alterada para que você consiga avançar nos estudos da forma mais tranquila possível.

Iniciaremos nosso estudo com a linha de comando do linux. Este passo é importante, para que você se familiarize com o Shell, aprendendo o básico sobre comandos de navegação de arquivos, redirecionamento de saída de comando, busca de arquivos e edição de texto através da linha de comando.

2.1 Como trabalhar na linha de comando

O Shell é uma interface entre o *kernel linux* e o usuário, uma camada interativa na qual você digita comandos. Além disso, o Shell é um ambiente de programação. Nele, criamos programas, que normalmente chamamos de Shell Scripts.

Existem diversas versões de Shell, sendo as mais conhecidas: Ksh, Sh, Bash, Fish e Dash.

O exame de certificação cobra apenas conhecimento no bash, que é o padrão de Shell nas distribuições de linux mais populares, como Ubuntu, Debian, Red Hat, Suse, Fedora, Centos e linux Mint.

Como saber qual Shell estou utilizando?

Como este livro tem como pré-requisito um conhecimento básico de linux, vou presumir que você esteja com seu sistema operacional linux instalado e funcionando. Se estiver no ambiente gráfico, você pode abrir um terminal de comando pelo seu menu iniciar, ou abrir o ambiente modo-texto, com as teclas: `<ctrl + alt + f2>` .

No ambiente de linha de comando, vamos iniciar nosso aprendizado com o comando `echo` . Este comando é comum em linguagens de programação e seu objetivo é imprimir na tela o resultado de algo que você venha a digitar. Por exemplo:

```
$ echo "linux Ubuntu"
linux Ubuntu
```

Não digite o sinal de cifrão `$` , ele representa que meu Shell está aberto com um usuário comum. Em alguns casos, teremos que executar os comandos como um usuário administrador, neste caso o símbolo será `#` , o sustenido. Para alterar de um usuário comum para o usuário administrador, também conhecido como root, executamos o comando:

```
$ su root
```

Será necessária a sua senha de administrador. Se você usa Ubuntu linux ou outra distribuição que não tem o usuário root disponível por padrão, execute o comando:

```
$ sudo passwd root
```

Informe a senha do seu usuário e depois informe uma senha para o seu usuário root, que agora estará disponível. Como usuário root, vamos descobrir qual Shell estamos utilizando em nosso sistema. Para isso, vou verificar a variável de sistema chamada `SHELL` que mantém esta informação. O comando é:

```
# echo $SHELL
/bin/bash
```

Observe que a saída foi `/bin/bash`, o que significa que estamos com o Shell correto para prosseguirmos nossos estudos. Se você não estiver executando o bash como padrão, é só chamá-lo, executando:

```
# bash
```

Mas, afinal, o que é um comando?

Um comando pode ser um programa externo (instalado no linux) ou um comando interno (do próprio Shell). Verificamos essa informação com um comando chamado `type`, exemplo:

```
# type echo
echo is a shell builtin
```

Outro exemplo:

```
# type tar
tar is /bin/tar
```

Observe a saída do primeiro comando que executamos, `type echo`. Ela informa que o comando `echo` é um "shell builtin" ou seja, um comando interno do Shell. Já para o segundo comando que digitamos, `type tar`, o resultado é um caminho para o comando executável do utilitário `tar`, o que significa que o `tar` é um comando externo.

Como o Shell sabe localizar um comando externo?

São tantos comandos no universo linux, que nos perguntamos: "Como o Shell sabe como encontrar e executar os comandos sempre que você os digita na linha de comando?" A resposta é simples:

Existe uma variável do Shell que guarda o caminho dos executáveis, chamada `PATH`. Você pode verificar o seu conteúdo executando:

```
# echo $PATH
/usr/local/bin;/usr/bin;/bin;/usr/sbin;/sbin
```

Observe que, na saída do comando, é exibida uma série de diretórios. Esses diretórios são os que armazenam os binários, que são comandos executáveis do linux. Assim sendo, quando você chama um comando, seu sistema o procura em cada um desses diretórios e, quando o encontra, ele é executado.

Mas e quando o comando não está no PATH?

Quando você digita um comando que não está em um dos diretórios referenciados na variável PATH, ele não será executado. Mas existem duas soluções para este caso: você pode executar o comando de dentro do seu diretório atual, ou adicioná-lo a algum diretório referenciado na variável PATH.

Digamos que você tem um programa de nome `sbrobols` dentro do diretório `/home/juliano/` , que é um diretório não referenciado na variável PATH. Para executá-lo, você primeiro deve acessar este diretório:

```
$ cd /home/juliano
```

Em seguida, executá-lo com algumas das opções possíveis:

```
$ bash sbrobols
```

Ou

```
$ ./sbrobols
```

Se você desejar executar esse programa, sem acessar seu diretório, deverá digitar o caminho completo:

```
$ /home/juliano/sbrobols
```

Variáveis de ambiente - local e global

Assim que você inicia o Shell, várias variáveis do seu sistema são carregadas automaticamente. Como o Shell é também um ambiente de programação, você pode declarar suas próprias variáveis. Para isso, coloque o nome da variável e o seu valor, por exemplo:

```
# linux="Eu uso Debian"
```

Para ver o valor da variável, digite:

```
# echo $linux
Eu uso Debian
```

Quando declaramos a variável do modo que realizei nesse exemplo, ela fica disponível apenas no seu Shell atual. Isso significa que, se você abrir um novo terminal de comandos, ela não vai existir, porque esta variável que declaramos é uma variável local.

Para que esta variável funcione também em outros terminais de comando, ela deve ser do tipo global. O comando que transforma uma variável local em global é o `export` , então realizamos:

```
# export linux
```

É possível já criar a variável como global, neste caso é só usar o `export` à frente do nome da variável no ato da criação:

```
# export linux="Eu uso Debian e Ubuntu"
```

Sendo assim, essa variável de nome `linux` vai estar disponível em qualquer terminal de comandos do seu sistema.

Ver todas as variáveis

Para visualizar todas as variáveis disponíveis em nosso sistema operacional linux, inclusive as que criamos, usamos o comando `set` . Exemplo:

```
# set
```

Como se trata de muitas variáveis, a saída do comando será grande. Você pode subir a tela do terminal com as teclas `<shift> + <Page Up>` e descer com `<shift> + <Page Down>` . Outra forma, mais eficaz, é usar um comando paginador de tela. Este comando lhe permite descer a tela com as setas direcionais. O comando padrão para paginar tela é o `less` , execute-o assim:

```
# set | less
```

Observe que o resultado da saída do comando será em partes, o que facilita sua leitura.

Ver apenas as variáveis globais

Diferente do comando `set` , que mostra todas as variáveis do sistema operacional, o comando `env` vai oferecer como saída apenas as variáveis globais. Ou seja, variáveis criadas com o comando `export` . Lembre-se de paginar a saída com o auxílio do comando `less` :

```
# env |less
```

Como remover uma variável

Antes de eu mencionar o comando de remoção de variável, fica um alerta: *Não remova as variáveis do sistema*. Algumas são necessárias para o funcionamento correto das aplicações e do próprio Shell. Utilize este comando para remover suas próprias variáveis.

O comando é `unset` mais o nome da variável:

```
# unset linux
```

Nesse exemplo, removi a variável `linux` , que tinha criado no exemplo de declaração de variáveis.

Comandos de modo sequencial

O mais interessante de se trabalhar com a linha de comando linux é a possibilidade de otimizarmos nossas tarefas. Para isso, podemos deixar uma lista de comandos para o sistema realizar enquanto degustamos um saboroso cafezinho.

Vou dar um pequeno exemplo de três comandos que eu desejo que sejam executados em sequência: `clear` , que limpa a tela, `date` , que mostra a data e a hora, e o comando `ls` , que vai mostrar o conteúdo do meu diretório atual. Segue:

```
# clear; date; ls
```

Nesse exemplo, os três comandos serão executados. Esta sequência usando ponto e vírgula não possui condição, ou seja, caso o comando falhe ou não, o próximo é executado normalmente.

Os próximos exemplos de comandos em sequência já possuem uma condição. Se você desejar que o segundo comando só seja executado se o primeiro tiver sucesso, faça:

```
# ls /tmp/qualquercoisa.txt && echo $SHELL
```

Se você não possuir o arquivo `/tmp/qualquercoisa.txt` , sua sequência de comandos será finalizada e o Shell retornará para a entrada padrão. Outra condição seria usando o `||` (dois pipes). Neste caso, o segundo comando só é executado quando o primeiro retorna com erro.

Exemplo:

```
# ls /tmp/qualquercoisa.txt || echo $SHELL
```

Como eu não tenho o arquivo `/tmp/qualquercoisa.txt` , somente o comando `echo $SHELL` será executado.

Navegar pelos comandos - histórico

Dentro do seu prompt de comandos, você poderá navegar pelos últimos comandos digitados, usando as teclas direcionais para cima

e para baixo. Outra opção é através do comando `history` , que mantém um histórico dos comandos digitados. Exemplo:

```
# history
1 uname -a
2 echo "linux"
3 clear
4 pwd
```

Observe na saída do comando `history` o número da linha. Com este número, você poderá executar diretamente um comando usando uma exclamação mais o número da linha, por exemplo:

```
# !3
```

Se observar na saída do meu comando `history` , a linha três corresponde ao comando `clear` . Isso significa que este é o comando que será executado.

Para executar o último comando digitado em seu terminal, digite:

```
# !!
```

Limpar arquivo de histórico

Se você deseja limpar seu histórico de comandos, utilize o comando:

```
history -C
```

Os comandos do `history` ficam armazenados em um arquivo, e para localizar este arquivo execute:

```
$ echo $HISTFILE
HISTFILE=/home/juliano/.bash_history
```

A variável `HISTFILE` informa o local deste arquivo. Você poderá visualizá-lo com o comando `cat` , por exemplo:

```
$ cat /home/juliano/.bash_history
```

Você também poderá remover este arquivo, para garantir mais segurança na exclusão do seu histórico de comandos:

```
$ rm /home/juliano/.bash_history
```

Obter ajuda

Praticamente todos os comandos têm um manual de referência, que podem ser acessados pelo comando `man`. Por exemplo:

```
$ man ls
```

Alguns comandos, além da página completa de manual, possuem um guia de referência, que pode ser acessado com a sintaxe `--help`, por exemplo:

```
$ ls --help
```

A página de manual é sempre bem detalhada, com exemplos e referências externas com mais informações; já o guia de referência é simples, com uma definição básica do comando.

Os comandos whatis e apropos

Muitas vezes, você deseja obter uma descrição sobre o que certo comando faz. Para isso, utilize o comando `whatis`, como em:

```
$ whatis tar
tar (1)              - The GNU version of the tar archiving utility
```

Nesse exemplo, estou perguntando o que faz o comando `tar` e o sistema me informa que ele é um utilitário de arquivamento. Por outro lado, você pode desejar saber quais comandos exibem alguma informação do sistema, ou seja, buscar comandos pela descrição. Nesses casos, utilize o `apropos`:

```
$ apropos information
```

O retorno será uma lista de comandos que tenham em suas descrições a string "information". Já que estamos citando

informação do sistema, um excelente comando para visualizarmos a nossa versão do kernel linux e a arquitetura do nosso sistema operacional é o comando `uname` , exemplo:

```
$ uname -r
```

Nesse caso, a saída será a versão do seu kernel linux. Para ter uma saída mais completa, mostrando também a arquitetura do sistema, use o comando:

```
$ uname -a
```

Apelidos de comandos

Um apelido é um nome que você cria para facilitar a entrada de um comando. Você pode ver todos os apelidos disponíveis usando o comando `alias` :

```
$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal
|| echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-
9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

No meu caso, quando digito `ls` , na verdade, ele vai executar: `ls --color=auto` como está nessa saída. Você pode criar no seu Shell atual um apelido usando o `alias` , por exemplo:

```
$ alias lt='ls /tmp'
```

Agora, sempre que digitar o comando `lt` que acabei de criar, será executado na verdade o comando `ls /tmp` .

Conclusão

A grande dica para este capítulo é: "Faça atividades sobre tudo o que aprendeu até aqui, isso reforçará a informação no seu cérebro". Não adianta você passar horas na teoria e não praticar o que se aprende. Já vi muita gente lendo livros técnicos como este no ônibus, ou no metrô, e isso não é eficaz. O essencial é você lê-lo à frente do seu computador, e neste momento, fazer uma pausa na leitura e executar todos os comandos, não uma, mas diversas vezes.

Algo que sempre funciona comigo é criar perguntas sobre o que leio. Algo do tipo: "Preciso criar um apelido de comando para listar o diretório `/etc`, como devo proceder?". Isso vai exigir que você pense no que leu para formular a pergunta e depois, pensar na resposta.

O exame LPIC-1 poderá fazer até quatro perguntas sobre este tópico. Mas não se preocupe, as questões são múltipla escolha, o que já facilita bastante. Pratique e você vai se sair muito bem.

Capítulo 3
Arquivos de texto

Praticamente, toda a configuração de um servidor linux pode ser realizada através de arquivos de texto puro. Isto é, um arquivo com conteúdo sem formatação. Quando você cria um documento no Libreoffice Writer, por exemplo, este arquivo possui diversos códigos, que informam cor da letra, tipo de fonte, formatação da página, entre outros. Um texto puro, por sua vez, possui apenas o conteúdo (caracteres) e, por isso, sua leitura é possível em qualquer dispositivo computacional.

Neste capítulo, veremos como manipular arquivos de texto com o Linux através da linha de comando (CLI - *Command-line Interface*). Os comandos `cat`, `tail` e `less` são usados diariamente na administração de um servidor linux. Em seguida, abordaremos os comandos `sed`, `expand` e `sort` que complementam a parte de leitura e filtro de um arquivo de texto.

Um exemplo de arquivo de configuração no Linux é o `/etc/fstab`, onde todas as informações relacionadas aos discos que vão iniciar com o linux ficam armazenadas. Vejamos um exemplo:

```
# <device>        <dir>        <type>        <options>        <dump>
<fsck>
/dev/sda1         /            ext4          noatime          0      1
/dev/sda2         none         swap          defaults         0      0
/dev/sda3         /home        ext4          noatime          0      2
```

Observe que é uma informação em texto puro, de forma que qualquer um com privilégio de administrador de sistema pode abri-lo e editá-lo.

Alterá-lo sem conhecimento prévio poderia fazer seu sistema não mais iniciar. Por esse motivo, teste os comandos deste capítulo em seus próprios arquivos de texto, e nunca nos arquivos do sistema.

Aprenderemos sobre os principais arquivos de configuração mais à frente neste livro.

O primeiro comando que vamos utilizar é para a leitura de arquivos em texto, chamado `cat`. Eu sempre brinco em minhas aulas, chamando-o de comando do gatinho, assim você não se esquecerá mais dele.

Abra qualquer editor de texto puro, como o gedit, kedit, lifepad, nano ou VIM e escreva: "Eu uso linux" e salve em seu diretório de usuário com o nome `linux.txt`. Agora, dentro do terminal de comando digite:

```
$ cat ~/linux.txt
```

O comando `cat` vai retornar o conteúdo do arquivo na próxima linha do Shell, ou seja:

```
$ Eu uso linux
```

Os comandos que vamos estudar modificam a saída do texto na sua tela. Não modificam o conteúdo do arquivo.

Exemplo:

Gostaria de ver a saída do arquivo `linux.txt` mostrando o número da linha à frente? Execute o comando `cat` com a sintaxe:

```
$ cat -n ~/linux.txt
```

E se por acaso você tem um arquivo de texto com diversas linhas em branco, mas gostaria de ignorá-las, neste caso, execute outra sintaxe do comando `cat`:

```
$ cat -b ~/linux.txt
```

Existem comandos com dezenas de sintaxes diferentes, e por isso é muito importante sempre consultar a página de manual do comando, ou verificar se ele possui a sintaxe de ajuda:

```
$ cat --help
```

Outro comando interessante é o `tac` , que na verdade é um `cat` invertido, ou seja, ele vai exibir o conteúdo do arquivo do final para o começo.

```
$ tac ~/linux.txt
```

Para exibir por padrão as 10 primeiras linhas de um arquivo de texto, utilize o comando `head` , que lê o cabeçalho do texto. Exemplo:

```
$ head ~/linux.txt
```

Uma sintaxe interessante do comando `head` é a possibilidade de definir o número de linhas a ser exibido, exemplo:

```
$ head -n2 ~/linux.txt
```

No exemplo, somente as duas primeiras linhas do arquivo `linux.txt` serão exibidas.

Assim como temos um comando que lê o cabeçalho de um arquivo de texto, temos outro que lê o rodapé, chamado `tail` . Este comando lê, por padrão, as 10 últimas linhas de um arquivo:

```
$ tail ~/linux.txt
```

Assim como o comando `head` , podemos definir o número de linhas a ser exibido:

```
$ tail -n2 ~/linux.txt
```

O comando `tail` possui uma sintaxe interessante. Ela permite que o arquivo fique sempre aberto no Shell e, caso seu conteúdo seja atualizado, como um arquivo de log, você verá as alterações em tempo real. A sintaxe do comando é:

```
$ tail -f /var/log/auth.log
```

No exemplo, eu estou abrindo o arquivo de log que registra os usuários que estão logando e realizando logoff no sistema. Deixe aberto esse arquivo de log com o comando anterior e abra um novo ambiente com as teclas: `< ctrl + alt + f3 >` .

Realize alguns logins. Retorne para o ambiente em que está o arquivo aberto e verifique a mudança. Vale lembrar que você pode alterar entre diversos ambientes, que são os terminais virtuais. Na maioria das distribuições de linux, vão de `< f1 >` até `< f7 >`.

3.1 Paginação da saída de texto

Se o seu arquivo de texto é muito grande, com centenas ou até milhares de linhas, ao executar o comando `cat`, por exemplo, a saída na tela será rápida, mostrando todo o conteúdo, porém o cursor vai parar nas últimas linhas. Você pode até subir a tela do Shell usando as teclas: `< ctrl + alt + PageUp >` e descer usando `< ctrl + alt + PageDown >`, mas isso é ineficaz quando você procura por uma informação específica dentro deste arquivo. Para isso, a melhor opção seria usar um paginador de texto. Vejamos o comando `less`:

```
# less /etc/passwd
```

Com a seta para baixo ou para cima, navegamos linha a linha. Se apertamos `< enter >`, ele pula para a próxima página. Podemos fazer busca na saída usando barra (`/`), por exemplo, `/juliano`. Nesse caso, ele vai procurar na saída do comando a palavra "juliano" e, se tiver mais de uma, usamos a tecla `< n >`, de *next* (próximo) se digitarmos `< q >` ele fecha o paginador.

O `less` é muito utilizado para paginar a saída de outros comandos. Para que isso ocorra, usamos um caractere chamado pipe (`|`) e o comando `less`. Exemplo:

```
# cat /etc/passwd |less
```

Nesse exemplo, a saída do comando `cat` será paginada com o comando `less`.

3.2 Contar palavras, linhas e caracteres

O comando `wc` permite que você conte o número de palavras, caracteres ou linhas de um arquivo de texto. Usar o comando sem nenhuma sintaxe retorna essas três informações. Por exemplo:

```
$ wc /etc/passwd
```

A saída deste comando no meu caso foi:

```
$ 40   70 2252 /etc/passwd
```

O que significa: 40 linhas, 70 palavras e 2252 caracteres.

Podemos usar algumas sintaxes para definir melhor essas informações, retornando apenas o necessário. Para linhas, utilize:

```
# wc -l /etc/passwd
```

Somente caracteres:

```
# wc -c /etc/passwd
```

Somente palavras:

```
# wc -w /etc/passwd
```

Outro comando para visualização de linhas é o `nl` , similar ao `cat -b` , pois ele enumera as linhas ignorando as linhas em branco:

```
# nl /etc/passwd
```

3.3 Ordenar arquivos

O comando `cat` sem dúvida alguma é um dos comandos Unix/linux mais utilizados no dia a dia de um administrador de sistemas. Porém, em alguns casos, é necessário utilizarmos outros recursos. Vamos vê-los.

Quando desejamos visualizar um arquivo organizando-o em ordem alfabética, o comando indicado é o `sort` :

```
# sort /etc/passwd
```

Ele permite também ordenar de modo reverso:

```
# sort -r /etc/passwd
```

3.4 Ver entradas únicas em um arquivo

O comando `uniq` permite que você visualize arquivos de texto, excluindo entradas duplicadas. Como exemplo, crie em seu editor e salve como `nomes.txt` um arquivo com o conteúdo:

```
juliano
mario
mario
juliano
maria
joaquim
juliano
mario
```

Observe que temos os nomes "juliano" e "mario" repetido três vezes. Para evitar essas repetições na saída, utilize o comando:

```
uniq alunos.txt
```

A saída será:

```
juliano mario maria joaquim
```

Para visualizar quantas vezes existiram repetições, utilize:

```
uniq -c alunos.txt
```

Nesse caso, a saída é:

```
3 juliano 3 mario 1 maria 1 joaquim
```

3.5 Tabulação

Tabulações são marcas definidas através da tecla `< tab >` que alteram a posição do seu texto. O Shell possui alguns comandos que permitem remover estas tabulações, assim como adicioná-las.

Isso é feito com os comandos `expand` e `unexpand`. Em ambos os comandos, a opção `-t` permite que você defina a quantidade de tabulações. Exemplo:

```
$ expand ~/texto.txt
```

Nesse exemplo, sempre que ele encontrar uma tabulação no arquivo, ele trocará pelo equivalente a oito espaços. Você pode definir este número de espaços usando:

```
$ expand -t 2 ~/texto.txt
```

Nesse caso, cada tabulação será trocada por dois espaços. O inverso também é possível, ou seja, trocar os espaços por tabulações:

```
$ unexpand arquivo.txt
```

A diferença é que, se você usar o comando `unexpand` sem nenhuma sintaxe, ele trocará apenas o começo da linha e não o seu texto todo. Para alterar em todo o texto, use desta forma:

```
$ unexpand -a arquivo.txt
```

Se você desejar, por exemplo, que a cada 2 espaços ele transforme em um tab, use:

```
$ unexpand -t 2 arquivo.txt
```

3.6 Combinar arquivos através de um índice

Pense na seguinte situação: seu aplicativo gerou dois relatórios. O primeiro possui o nome de todos os alunos da sua escola, e o segundo possui suas notas. Podemos unir os dois arquivos, gerando um único documento de forma simples através da linha de comando do linux. Para treinarmos, crie um arquivo com o seguinte conteúdo:

```
1 Maria
2 Joao
3 Lucas
```

Salve como `alunos.txt` dentro do seu diretório pessoal. Agora crie outro arquivo com o seguinte conteúdo:

```
1 10
2 5
3 3.8
```

Salve como `notas.txt` dentro de seu diretório pessoal. Execute agora:

```
$ join ~/alunos.txt ~/notas.txt
```

Sua saída será:

```
1 juliano 10
2 maria 5
3 lucas 3.8
```

O comando `join` realiza a junção de dois arquivos. Porém, eles devem ter um índice em comum, que, no nosso caso, foram os números em sequência ao início de cada linha. Poderia ser uma sequência alfabética, números romanos etc.

3.7 Dividir um arquivo

Quando temos um arquivo longo, podemos dividi-lo usando o comando `split`. No exemplo a seguir, vamos dividir o nosso arquivo `texto.txt`, que contém dezenas de linhas, em arquivos diversos com apenas três linhas de conteúdo cada:

```
$ split -l3 ~/texto.txt
```

Observe no seu diretório atual a criação de diversos arquivos com os nomes: xaa, xab, xac etc., cada um com 3 linhas de texto.

3.8 Apagar e substituir caracteres

O comando `tr` é usado para substituir caracteres. Como esse comando é apenas um filtro para a saída de outros comandos, devemos utilizá-lo com o pipe (`|`). Exemplo:

```
# cat /etc/passwd | tr a-z A-Z
```

Nesse exemplo, vamos alterar toda a saída do arquivo `/etc/passwd` para caracteres maiúsculos. O inverso também é possível. Caso tenha um texto escrito em maiúsculo, você pode usar:

```
$ cat texto.txt | tr A-Z a-z
```

Você pode realizar outras alterações, como:

```
$ cat /etc/passwd | tr ei EI
```

Nesse caso, ele transforma somente as saídas (`ei`) minúsculo para (`EI`) maiúsculo.

Outro exemplo interessante é a substituição de palavras em um arquivo:

```
$ cat /etc/passwd | tr juliano Grubelilo
```

Nesse caso, toda palavra "juliano" será alterada para "Grubelilo".

Para excluir um caractere ou uma palavra, use a opção `-d` , por exemplo:

```
$ cat /etc/passwd | tr -d juliano
```

Nesse caso, todas as palavras "juliano" serão excluídas. Mas vale lembrar que esses comandos não estão alterando o arquivo, somente trabalhando com as informações de saída, ou seja, com o que você visualiza na sua tela. No caso desse exemplo, o arquivo `/etc/passwd` não foi modificado.

3.9 Formatar uma saída de texto

Você pode limitar o tamanho das linhas em seu texto com o comando `fmt` . Isso é útil para uma melhor visualização na tela e em alguns casos para até mesmo a impressão de um documento. Exemplo:

```
$ fmt -w 20 /etc/passwd
```

Nesse exemplo, estou limitando a linha do arquivo `/etc/passwd` em 20 caracteres. Além do `fmt` , que formata a saída do texto, temos o comando `pr` que permite gerar uma saída pronta para ser impressa.

```
$ pr texto.txt
```

A saída do comando `pr` é interessante, porque ele adiciona um cabeçalho com o nome do arquivo, data e número da página. Você pode, inclusive, editar este cabeçalho de saída:

```
$ pr -h "Curso prof. Juliano Ramos" arquivo.txt
```

Agora, em todas as páginas do `arquivo.txt` , teremos como cabeçalho a informação: "Curso prof. Juliano Ramos".

Para finalizar este capítulo, vamos abordar mais dois comandos exigidos pela certificação LPI: `cut` e `sed` .

O comando `cut` recorta partes de um texto. Por exemplo, se você deseja obter na saída apenas os três primeiros caracteres do arquivo:

```
cut -c1,2,3 texto.txt
```

Você pode inclusive gerar um *range*:

```
cut -c1-5 texto.txt
```

O comando `sed` é uma poderosa ferramenta que tem como objetivo substituir, procurar e deletar conteúdo de um texto. Para a prova de certificação é fundamental conhecer as sintaxes de substituição e de exclusão:

```
sed 's/ramos/oliveira/' alunos.txt
```

Nesse exemplo, estamos substituindo a primeira ocorrência da palavra "ramos" pela palavra "oliveira". Para substituir em todo o texto, utilize:

```
sed 's/ramos/oliveira/g' alunos.txt
```

Já para deletar uma linha, utilize a sintaxe:

```
sed '3,5 d' alunos.txt
```

Nesse caso, o `sed` vai apagar as linhas 3 e 5 do arquivo `alunos.txt`. Você pode definir também que, sempre que uma ocorrência aparecer, a linha seja excluída. Por exemplo:

```
sed '/juliano/d' alunos.txt
```

Agora, toda vez que a palavra "juliano" aparecer dentro de `alunos.txt`, a linha será excluída.

Conclusão

A prova de certificação LPI é múltipla escolha. Com o passar do tempo, você vai observar um padrão em muitas sintaxes de comando, de modo que, no dia do exame, será fácil descartar as

respostas erradas. Não é necessário estudar a página de manual de cada comando - isso seria loucura, já que alguns comandos possuem dezenas de sintaxes. Mas é importante compreender o funcionamento padrão de cada um. Nosso livro sempre vai aproximá-lo ao máximo do exame de certificação e, por isso, revise sempre os capítulos à frente do computador. Mais do que ler, é fundamental que você pratique.

Capítulo 4
Gerenciamento de arquivos

Outro tema cobrado na prova de Certificação LPI é o entendimento da estrutura de arquivos e diretórios. Desde que o GNU/linux foi criado, muito se tem feito para seguir um padrão em relação à organização e à padronização.

O primeiro esforço para a padronização de sistemas de arquivos do GNU/linux foi o **FSSTND - Filesystem Standard**, lançado no ano de 1994.

Atualmente é o **Filesystem Hierarchy Standard ou FHS** que determina o padrão da estrutura de diretórios do GNU/linux. Sua versão atual é a **2.3**, anunciada em 29 de janeiro de 2004.

O FHS é mantido pelo Free Standards Group, uma organização sem fins lucrativos formada por importantes empresas de Hardware e Software, como HP, Red Hat, IBM e Dell.

Neste capítulo, você vai aprender a manipular arquivos e diretórios no linux. Dominar este conteúdo é importante para o exame de certificação e para o trabalho diário com um ambiente linux.

4.1 Estrutura de diretórios

A estrutura de diretórios também é conhecida como "Árvore de diretórios" porque ela tem a forma de uma árvore. Um diretório é como uma gaveta de armário, sendo que cada gaveta guarda, normalmente, um tipo diferente de roupa, enquanto cada diretório guarda um certo tipo específico de arquivo. O arquivo pode ser um texto, uma imagem, um vídeo, uma planilha entre outros, identificados por nomes para que sejam localizados com facilidade.

Um detalhe importante a ser observado é que o GNU/linux é *case sensitive*, isto é, ele diferencia letras maiúsculas nos arquivos e diretórios. Sendo assim, um arquivo chamado "**Arquivo**" é diferente de outro chamado "**ARQUIVO**" ou de outro chamado "**arquivo**".

Para iniciarmos nosso estudo, executaremos o comando `ls`, que lista arquivos e diretórios. Partiremos do diretório raiz do sistema, de modo que através dele visualizaremos todos os outros, compreendendo a estrutura hierárquica de diretórios do GNU/linux.

```
$ ls /
```

Com isto, você verá a estrutura de diretórios do GNU/linux. De acordo com o padrão FHS, essa estrutura deve conter obrigatoriamente 14 diretórios, que vamos especificar a seguir.

Diretório /

O diretório raiz do GNU/linux é representado por uma `/` (barra). É no diretório raiz que ficam todos os demais diretórios do sistema. Esses diretórios, que vamos conhecer agora, são chamados de subdiretórios pois estão dentro de `/`.

Diretório /bin

Guarda os comandos essenciais para o funcionamento do sistema. Esse é um diretório público, sendo assim, os comandos que estão nele podem ser utilizados por qualquer usuário do sistema. Entre esses comandos estão:

- `/bin/ls` : lista arquivos e diretórios.
- `/bin/cp` : copia arquivos e diretórios.
- `/bin/mkdir` : cria diretórios no sistema.
- `/bin/cat` : visualiza arquivos de texto.

Diretório /boot

Aqui estão os arquivos necessários à inicialização do sistema e o gerenciador de boot. O gerenciador de boot é um programa que lhe permite escolher os seus sistemas operacionais na inicialização do seu computador. Quando o computador é ligado, uma lista aparece para que você escolha entre diferentes versões do kernel linux ou até mesmo outros sistemas operacionais que você possua, como o Microsoft Windows.

Diretório /dev

Aqui ficam todos os **arquivos de dispositivos** (hardware do seu computador). O linux faz a comunicação com esses periféricos por meio de "links" especiais que ficam armazenados neste diretório, facilitando o acesso a eles.

Diretório /etc

Neste diretório estão os arquivos de configuração do sistema, tais como: scripts de inicialização do sistema, tabela do sistema de arquivos, configuração padrão para logins dos usuários, entre outros.

Diretório /lib

Aqui estão as bibliotecas compartilhadas e módulos do kernel. As bibliotecas são funções que podem ser utilizadas por vários programas. Cada kernel tem seus próprios módulos, que ficam em: `/lib/modules/versao-do-seu-kernel/kernel .`

Diretório /media

É um ponto de montagem para dispositivos removíveis, tais como: HD, CD, DVD entre outros. Um ponto de montagem é um diretório que aponta para a partição física do disco, permitindo que você

acesse seus arquivos e diretórios. Mais à frente no livro, abordaremos em detalhes a criação de partições e como acessá-las.

Diretório /mnt

Este diretório é utilizado para montagem temporária dos sistemas, ou seja, para acessar um compartilhamento de arquivo em uma rede que depois não mais acessaremos, ou um HD externo que logo será desconectado do sistema.

Diretório /opt

Normalmente é utilizado por programas proprietários ou que não fazem parte oficialmente da distribuição.

Diretório /sbin

Guarda os comandos utilizados para inicializar, reparar, restaurar e/ou recuperar o sistema. Isso quer dizer que esse diretório contém comandos essenciais. Esse diretório é de uso do Administrador - root.

Diretório /srv

Diretório para dados de serviços fornecidos pelo sistema, cuja aplicação é de alcance geral, ou seja, os dados não são específicos de um usuário, por exemplo:

`/srv/ftp` (Servidor FTP): diretório usado para usuários anônimos que se conectam a um servidor FTP.

Diretório /tmp

Diretório para armazenamento de arquivos temporários.

Diretório /usr

Contém programas que não são essenciais ao sistema e que seguem o padrão GNU/linux, como navegadores, gerenciadores de janelas, entre outros.

Diretório /var

Contém arquivos de dados de variáveis e arquivos de logs.

Diretório /proc

É um diretório virtual, ou seja, ele está armazenado na memória RAM e não no disco, e é mantido pelo kernel. Nele, encontramos configurações do sistema, dados estatísticos, dispositivos já montados, interrupções, endereços e estados das portas físicas além de dados sobre as redes.

Diretório /sys

É um primo do diretório `/proc`. Dentro dele, podemos encontrar quase o mesmo conteúdo do outro, mas de uma forma mais organizada para os administradores, já que ele estrutura melhor os subdiretórios. Para nós, administradores linux, não existe diferença entre esses diretórios, é mais uma questão de gosto pessoal.

Diretório /home e /root

Esses diretórios podem estar disponíveis no sistema, mas não precisam obrigatoriamente possuir esses nomes. Por exemplo, o diretório `/home` poderia se chamar `/usuarios`, que não causaria nenhum impacto na estrutura do sistema.

No diretório `/home`, temos os diretórios dos usuários, por exemplo, `/home/juliano` e `/home/jaqueline`. E, dentro desses diretórios, `juliano` ou `jaqueline`, temos os arquivos pessoais deste usuário, tais como: músicas, filmes, documentos de texto, planilhas entre outros. No

diretório `/root` , temos os arquivos pessoais do usuário administrador do sistema.

4.2 Navegando pelos diretórios

Como usuário `root` (administrador), você pode acessar qualquer diretório do sistema, colocando o comando `cd` e o nome do diretório. Os usuários comuns possuem restrições de acesso a diretórios que possam comprometer a segurança do sistema.

Quando acessamos um diretório, por exemplo:

```
# cd /usr/share/
```

Podemos subir um nível de subdiretório usando o comando:

```
# cd ..
```

Observe que você agora está no diretório `/usr/` e não mais em `/usr/share` . Para voltar dois níveis de diretórios, execute:

```
# cd ../../
```

Um atalho interessante é o `~` til, que vai movê-lo diretamente para seu diretório `/home/usuario` . Exemplo:

```
# cd ~
```

Para listarmos os arquivos que estão em um diretório, usamos o comando `ls` :

```
# ls /home/juliano
```

Esse comando vai listar todos os arquivos do diretório `/home/juliano` exceto os arquivos ocultos. Arquivos ocultos no linux começam com

. (ponto), por exemplo: `.arquivo-oculto.txt` . Para listar esses arquivos, execute a opção `-a` :

```
# ls -a /home/juliano
```

Outra opção comum é `-l` , que mostra uma lista detalhada. Podemos usar em conjunto com o `-a` para um melhor resultado:

```
# ls -la /home/juliano
```

Outra opção interessante é usar o `-h` , que mostrará os tamanhos dos arquivos em Byte, KByte, MByte etc.

```
# ls -lha /home/juliano
```

Para listar somente arquivos que tenham a extensão `.txt` , por exemplo, usamos o caractere curinga asterisco. Neste caso, acesse o diretório em que deseja listar esses arquivos e execute o comando:

```
# ls *.txt
```

Outra situação interessante é a possibilidade de definir quais arquivos listar. Por exemplo, digamos que você tenha uma lista com vários arquivos chamados aula1, aula2, aula3 etc. Você pode definir quais listar usando:

```
# ls -l aula[123]
```

Para mostrar todos, menos o aula1, aula2 e aula3, você pode usar:

```
# ls -l aula[!123]
```

O ponto de exclamação (`!`) é uma negação. Ou seja, ele listará todos, exceto os que foram sinalizados com a exclamação.

Cópia de arquivos

O comando `cp` permite que você copie arquivos, por exemplo:

```
# cp origem destino
```

ou

```
# cp /etc/passwd /tmp
```

Outra opção é ver os detalhes durante a cópia, com a qual ficará passando na tela todo o processo de cópia. Exemplo:

```
# cp -v /etc/passwd /tmp
```

A opção `-i` é interessante, porque o sistema perguntará sempre antes de sobrescrever um arquivo se isso deve ou não ser realizado:

```
# cp -i /etc/passwd /tmp
```

Cópia de diretórios

O comando `cp` não copia por padrão diretórios, mas podemos resolver isso usando a opção de recursividade:

```
# cp -r diretorio /tmp/
```

Mover e renomear um arquivo

Usamos o comando `mv` para mover um arquivo e também para renomeá-lo. Exemplo:

```
# mv diretorio /tmp
# mv arquivo /tmp
# mv arquivo arquivo2
```

Observe que não usamos `-r` (recursivo) no `mv` para diretórios.

Criando arquivo de texto vazio

Usamos o comando `touch` para criar um arquivo vazio de texto, por exemplo:

```
# touch arquivo.txt
```

Quando você executa o comando `ls -l` , consegue visualizar a hora e data de criação e modificação de um arquivo. O comando `touch` permite que você altere essas informações. Para mudar a data do último acesso, colocando a data e hora atual, execute:

```
# touch -a arquivo.txt
```

Para alterar a data da última modificação:

```
# touch -m arquivo.txt
```

Podemos alterar a data de criação de um arquivo, para qualquer data e hora. Neste caso, alterei para o ano de 2017, mês 01, dia 01, hora 10:00. Veja:

```
# touch -t 201701011000 arquivo.txt
```

Remover arquivos e diretórios

O comando é o `rm` :

```
# rm arquivo
```

Para remover diretório, usamos a recursividade:

```
# rm -r diretório
```

Remover diretório vazio

Para remover diretório vazio, pode-se usar o comando `rmdir` .

```
#rmdir diretorio-vazio
```

Criar um diretório

Para a criação de um diretório, usamos o comando `mkdir` :

```
# mkdir diretorio
```

Se você desejar criar um diretório e, dentro, um subdiretório, ou até mesmo uma árvore inteira de subdiretórios, use a opção `-p` ou `--parents`, que tem mesmo efeito. Por exemplo:

```
# mkdir -p diretorio/diretorio2/diretorio3
```

Procurar arquivos e diretórios

Para procurarmos arquivos e diretórios usamos o comando `find`, exemplo:

```
# find /home -name aula1.txt
```

Esse comando vai procurar o `arquivo1.txt` dentro do diretório `/home`. Para vermos todos os arquivos que começam com "aula" neste diretório, usamos o caractere curinga asterisco. Por exemplo:

```
# find /home -name aula*
```

Para restringir a busca por somente arquivos criados por um determinado usuário, usamos:

```
# find /home -user juliano -name aula1
```

É possível também procurarmos por arquivos modificados nos últimos dias. Nesse caso, usamos:

```
find ./ -ctime -1
```

Estou procurando a partir do meu diretório atual (`./`) arquivos modificados no último dia.

Agrupando arquivos

O comando `tar` agrupa diversos arquivos e diretórios em um único arquivo, o que é muito interessante para criarmos backup. Veja um exemplo:

```
# tar cpvf backup.tar /tmp
```

Nesse caso, vamos obter todo o conteúdo do diretório `/tmp` transformando-o em um único arquivo chamado `backup.tar`. Vamos entender melhor as opções que usamos no comando:

- `c` : criar o arquivo.
- `p` : manter as permissões dos arquivos.
- `v` : modo verbose - Detalhes de todo o processo da criação do pacote é exibido na tela.
- `f` : apontar o arquivo (sempre será utilizado).

Não estamos compactando, ou seja, diminuindo o tamanho dos arquivos, estamos apenas agrupando.

Compactando arquivos

Diferente de agrupar, compactar significa que vamos reduzir o tamanho dos arquivos e diretórios. No linux, podemos utilizar diversos softwares de compactação. Os mais comuns são o `gzip`, `bzip2` e `xz`.

Para iniciar, vamos começar com o `gzip`, usando aquele arquivo que agrupamos. Execute o comando:

```
# gzip backup.tar
```

Observe que ele gera um novo arquivo chamado `backup.tar.gz`, que significa que está compactado. Para descompactar, usamos:

```
# gunzip backup.tar.gz
```

Agora a descompactação foi realizada e voltamos ao original `backup.tar`. Se você deseja criar um arquivo compactado e manter uma cópia do `backup.tar`, execute o comando:

```
# gzip -k backup.tar
```

O `bzip2` é uma outra aplicação de compactação. Seu uso é tão simples quanto o `gzip`. Veja:

```
# bzip2 -k backup.tar
```

Ele vai criar o arquivo `backup.tar.bz2`. Para descompactar, use:

```
# bunzip backup.tar.bz2
```

O `xz` não é tão utilizado no mundo linux, porque é mais lento que os outros na compactação e descompactação. Seu uso seria assim:

```
# xz backup.tar
```

Para descompactar:

```
# unxz backup.tar.xz
```

Podemos usar também para descompactar a opção `-d`: `gzip -d`, `bzip2 -d`, `xz -d`. Vale lembrar que eles compactam qualquer arquivo e não somente arquivos com a extensão `.tar` como utilizei nos exemplos.

Usando somente o tar

É possível usar o comando `tar` para criar arquivos compactados e descompactar. Por exemplo, para já criar um arquivo compactado em `gzip`, utilize:

```
# tar zcvpf backup.tar.tgz /tmp
```

Para já criar compactado em `bzip2`:

```
# tar jcvpf backup.tar.tbz /tmp
```

Para `xz`, usamos:

```
# tar Jcvpf backup.txz /tmp
```

Para verificar como foi compactado determinado arquivo (caso não tenha extensão), use o comando `file`:

```
# file nome_do_arquivo
```

Para descompactar qualquer um deles, é só alterar `c` para `x`. Por exemplo:

```
# tar zxvf backup.tar.tgz
```

Usando o cpio

O `cpio` é semelhante ao `tar`, porém, ele necessita de uma lista de arquivos como entrada. Por padrão, ele é usado com o comando `find`. Veja um exemplo:

```
# find /tmp | cpio -o > backup.cpio
```

Neste caso, estamos criando o backup do diretório `/tmp` com o nome `backup.cpio`. Para desagrupar, usamos:

```
# cpio -i < backup.cpio
```

Copiar uma partição inteira

O comando `dd` permite uma cópia bit a bit de uma partição ou até mesmo de um disco inteiro. Você pode verificar as partições que você tem no sistema com o comando:

```
# df -h
```

Para fazer uma cópia exata de um disco, use:

```
# dd if=/dev/sda of=/dev/sdb
```

Nesse exemplo, criamos uma cópia exata de um disco para outro. Isso é importante, por exemplo, no caso de um backup de servidor. Se você quiser fazer uma imagem (`.iso`) de um disco (DVD, Blu Ray), utilize o comando:

```
# dd if=/dev/sr0 of=imagem.iso
```

Conclusão

Este capítulo é de peso 4 no exame da LPI, o que equivale a quatro questões deste tema na prova. O mais comum são as questões com foco em compactação/descompactação de arquivos, principalmente, o conteúdo sobre o comando `tar` e `gzip`.

Capítulo 5
Redirecionamento de entrada e saída

A Certificação Linux LPI tem como objetivo formar administradores de sistemas Linux. Um administrador deve dominar diversos comandos e saber como filtrar a saída, armazenando em arquivos que possam ser enviados para outros administradores ou mesmo para serem armazenados como backup.

No linux, temos uma entrada padrão que identificamos como sendo de nome: **STDIN** número **(0)** - que são as informações que chegam ao seu computador pelo seu teclado. Temos uma saída padrão **STDOUT** de número **(1)** - que são as informações que saem do seu computador pela tela do monitor e, por fim, uma outra opção, que são saídas de erros de comandos, que também utilizam o monitor - sua nomenclatura é: **STDERR** de número **(2)**.

É possível redirecionar a saída de um comando para um arquivo de texto comum em vez do monitor, deste modo, você pode arquivar toda a saída de um comando em um documento, o qual, você pode guardar para estudar, encaminhar para um colega administrador, ou mesmo salvar como backup ou histórico de sua ação no servidor. Para isso, usamos o sinal de maior (`>`). Por exemplo:

```
# ls -l /tmp > saida.out
```

Nesse exemplo, em vez de o resultado do comando `ls` ser a tela do computador, estamos enviando essa saída para um arquivo, chamado: `saida.out` . Se você usar o comando `cat` ou o `less` neste arquivo, perceberá que o conteúdo resultante do comando está no arquivo `saida.out` .

```
# cat saida.out
```

Atenção, se você executar novamente o comando:

```
# ls -l /tmp > saida.out
```

e o conteúdo do seu diretório `/tmp` tiver tido alguma mudança, o arquivo `saida.out` será sobrescrito. Isso porque o `>`, além de redirecionar o comando, sobrescreve. Veja:

```
# df -h > saida.out
```

Nesse caso, estamos reescrevendo o arquivo `saida.out` colocando o resultado do comando `df -h`.

Para adicionar conteúdo a esse arquivo e não o reescrever, use `>>`. Por exemplo:

```
# ls /tmp >> saida.out
```

Nesse caso, o conteúdo de saída do comando será adicionado no final do arquivo.

5.1 Redirecionamento do erro

Quando você executa um comando que retorna um erro no Shell, este erro aparecerá na sua tela.

Por exemplo, tente listar um arquivo que não existe em seu computador:

```
# ls /tmp/arquivo-inexistente
ls: não é possível acessar /tmp/arquivo-inexistente: Arquivo ou diretório não encontrado
```

Se você redirecionar a saída desse comando, que não existe para um arquivo, usando o redirecionador `>`, este arquivo será vazio, pois o redirecionador comum só redireciona os conteúdos (`stdout`) e não as saídas de erros (`stderr`). Exemplo:

```
# ls /tmp/arquivo-inexistente > saida_ls
# cat saida_ls
#
```

Isso acontece porque o comando não produziu nenhum resultado, apenas a mensagem de erro. Porém, a mensagem de erro não será adicionada ao arquivo, porque, para isso, você deve colocar o número correspondente à saída de erro, que é dois (2), ficando desta forma:

```
# ls /tmp/arquivo-inexistente 2> saida_ls
```

Observe agora que temos um arquivo chamado `saida_ls` com a informação do erro. Visualize-o usando o comando:

```
# cat saida_ls
ls: não é possível acessar /tmp/arquivo-inexistente: Arquivo ou diretório não encontrado
```

É possível também criar um arquivo com as duas saídas, uma para o padrão **STDIN1**, que é a saída correta de um comando, e outra para **STDERR2**, que é a saída de erro, caso exista algum erro.

Como exemplo, vou usar dois arquivos, um que existe no diretório e outro que não existe. Vou criá-lo com o comando `touch` e depois executar uma listagem no arquivo que existe e outra no arquivo que não existe.

Observe:

```
# touch /tmp/existo
# ls -l /tmp/{existo,nao-existo} > saida-ls.out 2> saida-ls-erro.out
```

Esse comando gerou dois arquivos, um com erro `saida-ls-erro.out` e o outro com a saída correta do comando `saida-ls.out`:

```
tux@giovanna /tmp> cat saida-ls.out
-rw-r--r-- 1 tux tux 15 Ago 30 18:25 /tmp/existo

tux@giovanna /tmp> cat saida-ls-erro.out
ls: não é possível acessar /tmp/nao-existo: Arquivo ou diretório não encontrado
```

Se desejarmos redirecionar a saída padrão e a saída de erro ao mesmo arquivo, utilizamos desta forma:

```
ls -l /tmp/{existo,nao-existo} > saida-ls.out 2>&1
```

5.2 Visualizando a saída e criando um arquivo

O comando `tee` permite que você visualize a saída do comando e ao mesmo tempo crie um novo arquivo com esta informação. Sua sintaxe é assim:

```
# ls -l /tmp | tee novo_arquivo.tmp
```

Veja que você teve a saída do comando, mas também terá um arquivo chamado `novo_arquivo.tmp` com esta informação.

5.3 Saída de um comando como entrada para outro comando

Para pegarmos a saída de um comando e usarmos como argumento para outro comando, usamos o comando `xargs` . Exemplo prático:

```
# find /etc -name "passwd" | xargs ls -l
```

No exemplo acima, ele pega o resultado da busca de arquivos com a string `"passwd"` dentro de `/etc` e usa como argumento para o comando `ls -l` .

5.4 Imprimindo o resultado de um comando

Podemos imprimir o resultado de um comando usando crase, veja:

```
# echo "A versão do kernel é: " `uname -r`
```

Outro modo é com cifrão e parênteses, por exemplo:

```
# echo "A versão do kernel é:" $(uname -r)
```

Conclusão

Este é um tópico de peso quatro (4) e um conteúdo relativamente simples. Atente-se às saídas padrões. Domine o que é STDIN1, STDOUT e STDERR. Fique alerta para a diferença entre `>` e `>>`, lembre-se que maior (`>`) redireciona e sobrescreve um arquivo e maior maior (`>>`) redireciona e adiciona conteúdo.

Capítulo 6
Gerenciamento de processos

Todo programa e script que você inicia no Linux é um processo. Todo processo tem um número de identificação chamado de PID - Process Identifier. Além do PID, temos o PPID - Parent Process ID, que é o processo pai dentro de uma hierarquia de processos. Exemplo: Se você abre o bash e depois executa uma aplicação como o editor vim, o processo PPID será o bash. O primeiro processo do Linux é o `INIT` e o PID dele é sempre ***1***. Este é o único processo que tem um PID de número fixo, os outros têm números variados.

Neste capítulo, vamos aprender como listar os processos ativos no sistema e enviar sinais para esses processos, que podem ser para reiniciá-los ou realizar sua paralização. Conhecer sobre gerenciamento de processos é de suma importância para o administrador de sistema. Você poderá, por exemplo, verificar qual processo está consumindo mais recurso da sua máquina, assim como finalizar um processo que esteja em erro.

6.1 Comando ps

O comando `ps` é usado para mostrar os processos da máquina. Ele tem diversas opções, mas usaremos as que possivelmente estarão no exame.

Ao executarmos apenas o comando `ps` , vamos visualizar apenas os processos do nosso usuário:

```
# ps
```

Para ver mais detalhes sobre os processos, usamos a opção `-u` , de usuário:

```
# ps -u
```

A sintaxe `-u` permite ao comando `ps` mostrar o PID e informações, como uso da CPU e da memória que esse processo está consumindo. Outra opção muito utilizada é o `-x` , que mostra processos que não foram iniciados diretamente em nosso terminal.

```
# ps -ux
```

Para visualizar todos os processos incluindo processos de outros usuários, usamos `-a` :

```
# ps -aux
```

Para ver em formato de árvore hierárquica de processos, podemos usar a opção `-f` :

```
# ps -auxf
```

Para ver a prioridade do processo, podemos usar a opção `-l` :

```
# ps -alx | less
```

Usamos muito o comando `ps` com o pipe (`|`) para filtrar uma saída, por exemplo:

```
# ps -aux | grep firefox
`
```

Neste exemplo, estou listando apenas os processos do browser Firefox.

6.2 Comando pstree

O comando `pstree` mostra os processos em formato de árvore hierárquica.

```
# pstree
```

Uma opção interessante é o `-p` , que mostra os números do PID:

```
# pstree -p
```

Observe que o `pstree` mostra que o PID 1 é o INIT. Como mencionei anteriormente, este é o único processo que tem um PID fixo.

O INIT era o sistema antigo de inicialização que hoje aponta para o systemD. Então de fato o processo 1 é o systemD. Mas, em uma questão no exame sobre qual seria o processo 1, a sua resposta seria INIT (por padrão).

6.3 Procurando pelo PID dos processos

Podemos usar o comando `pgrep` para encontrar o PID de um processo. Por exemplo:

```
# pgrep firefox
```

Para ver processo de um usuário específico, utilize:

```
# pgrep bash -u usuario
```

6.4 Comando top

O `top` é um dos principais comandos para monitorar o desempenho da máquina. Ao digitá-lo sem nenhum parâmetro, você terá uma tela interativa, com diversas informações sobre o equipamento, que incluem número de processos, uso de CPU, memória, entre outros.

Dentro do comando `top`, se você digitar `shift + M`, visualizará os processos que estão consumindo mais memória.

Para ordenar por uso de CPU, o comando é `shift + U`.

Para mostrar processos apenas de um usuário, digite `u` e o nome do usuário na caixa de diálogo. Quando você não define um usuário, o processo de todos é mostrado.

Para visualizar a saída do `top` sem usar tela interativa, use:

```
# top -b
```

Isso é útil para criação de log.

6.5 Matando processos travados

Existe uma tabela com vários sinais que você pode enviar para um processo. Você pode visualizá-la com o comando:

```
# kill -l
```

Estes sinais servem para reiniciar, parar e finalizar um processo que esteja com algum erro.

Entre os vários sinais, os que mais utilizamos é o `-15`, que finaliza um processo, e o `-9`, que força a finalização de um processo. Exemplo:

```
# kill -9 PID
```

6.6 Comando killall

O comando `killall` mata os processos com base nos nomes dos processos e não no seu número de PID. Exemplo:

```
# killall firefox
```

Isso vai matar todos os processos com nome `firefox`.

Observação: os processos possuem permissões. Apenas o root pode matar os processos de qualquer usuário. Um usuário comum não pode matar o processo de outro usuário comum.

6.7 Comando pkill

O `pkill` permite que você faça um filtro em processos pelo nome e pelo nome do usuário.

Exemplo: digamos que você está no servidor como root e há quatro usuários rodando o processo `firefox`. Para matar o processo apenas do usuário `juliano`, você executaria:

```
# pkill -9 firefox -u juliano
```

6.8 Controle de tarefas

Quando iniciamos um processo no terminal, o processo fica preso no comando, por exemplo:

```
 # nano
```

Você abre o editor `nano`, mas seu terminal não pode mais ser utilizado até o nano ser fechado. Chamamos isto de processo em *foreground*. Para executar um processo, liberando nosso terminal, usamos o que se chama processo em *background*. Nesse caso, basta colocar um `&` comercial ao final do comando:

```
# nano &
[1] 38
```

O editor nano vai ser aberto em segundo plano. Observe a saída do meu caso:

```
[1] 38
```

Ela representa o número do job em segundo plano `[1]` e o número do PID `[38]`.

O comando `jobs` mostra os processos em *background*:

```
# jobs -l
[1]+    38 Stopped (tty output)    nano
```

Se você desejar colocar o processo em *foreground*, você usa o comando `fg + número do Jobs`. No meu caso:

```
# fg 1
`
```

Conclusão

Domine o comando `ps` e as suas funções. Também, atente-se à diferença entre matar um processo (-9) e finalizar um processo (-15). É provável que você encontrará questões sobre manipulação da lista de jobs. Além disso, não confunda PID com número de jobs, que são processos que você coloca em *background*.

Capítulo 7
Pesquisar arquivos de texto com expressões regulares

A expressão regular é um conjunto de caracteres que define um padrão de texto. Trata-se de um assunto muito intenso, sendo uma grande área de estudo, no entanto, vamos focar o conteúdo apenas no que exige a certificação LPIC-1 101, que é o nosso objetivo.

Se você desejar se aprofundar no assunto, o livro mais indicado é o do Aurélio Jargas, *Expressões Regulares - Guia de Consulta Rápida*, publicado em 2001 pela Editora Novatec.

7.1 Comando grep

O comando `grep` pesquisa blocos de texto através de strings que você especifica. Por exemplo:

```
# grep juliano /etc/passwd
```

Estamos buscando pelo conteúdo `juliano` dentro do arquivo `/etc/passwd`. Tente realizar essa operação com seu nome de usuário. Vale ressaltar que a busca é *case sensitive*, ou seja, existe diferença entre letras minúsculas e maiúsculas.

Podemos utilizar junto com o comando `grep` a sintaxe `-c`, que vai contar quantas ocorrências existem no arquivo `/etc/passwd` com a string `juliano`:

```
# grep -c juliano /etc/passwd
```

Outra opção é `-i`, que vai ignorar o *case sensitive.*

```
# grep -i JULIANO /etc/passwd
```

A opção `-r` é recursiva. Ela procura nos diretórios atuais e também dentro dos subdiretórios a string especificada:

```
# grep -r juliano /etc/*
```

No exemplo, realizei a procura de `juliano` dentro dos diretórios e subdiretórios que estão em `/etc`.

A opção `-v` exclui uma determinada palavra na busca:

```
# grep -v juliano /etc/passwd
```

Nesse exemplo, o arquivo `/etc/passwd` é aberto, omitindo a linha que contém a string `juliano`.

Duas ou mais opções podem ser usadas em conjunto para exibirmos resultados diferentes. Para isso, realize:

```
# grep -vi JULIANO /etc/passwd
```

Observe que, nesse caso, o comando vai omitir (JULIANO) e (juliano), ou seja, ele exclui o *case sensitive*.

7.2 Comando egrep

Antes de conhecermos o `egrep`, vamos entender a diferença entre o `grep` e o `egrep`. O `grep` é um utilitário já embutido no Kernel para efetuar um filtro durante uma pesquisa. Por exemplo: durante esta pesquisa, quero que você me retorne somente os arquivos com a extensão `.doc`. Isso é possível com o `grep`.

O `egrep` é um atalho para o comando `grep -E`. A opção `-E` permite o uso de expressões regulares para filtragem de texto, como `+`, `|`, `()` etc.

Por exemplo, vamos buscar a palavra `"tux"` em um arquivo chamado `linux.txt` usando o `grep`:

```
grep "tux" arquivo.txt
```

Agora se desejarmos buscar duas palavras (tux e torvalds) neste arquivo, usamos o `egrep` :

```
egrep "tux|torvalds" arquivo.txt
```

Para trabalhar com expressões usando o `egrep` , usaremos a expressão entre aspas. Veja:

```
# egrep "b[aei]g" arquivo.txt
```

No exemplo, estamos procurando palavras como "bag", "beg" e "big" (palavras que comecem com b e terminem com g) dentro do `arquivo.txt` . Outra forma seria:

```
# egrep "b[a-u]g" arquivo.txt
```

Nesse exemplo, o comando vai mostrar letras de A a U (a, e, i, o, u).

Podemos definir também o começo da linha e o seu fim:

```
# egrep "^Linux" arquivo.txt
```

No exemplo acima, o comando retornará todas as palavras (linux) que estiverem no começo dos parágrafos. Para exibir as palavras (linux) no final dos parágrafos, use:

```
# egrep "Linux$" arquivo.txt
```

Exemplo prático:

Para ver todas as linhas com comentários, que estão sempre no começo do parágrafo, usamos:

```
# egrep "^#" arquivo.txt
```

Para ver todas as linhas sem os comentários:

```
# egrep -v "^#" arquivo.txt
```

Outro exemplo:

```
# egrep "b[a-i]g*" arquivo.txt
baseado no debian
bag
beg
big
```

O `*` após o caractere informa que, tanto faz se o caractere `g` aparece ou não, o conteúdo deve ser exibido. Observe no retorno (baseado no debian) que apareceu, isso porque tem (`bas`) da palavra baseado, mesmo que não tenha `g` .

Agora, vamos especificar um pouco mais:

```
# egrep "b[a-i]g*e" arquivo.txt
bigger
bege
```

Quando coloco `g*e` no exemplo acima, estou definindo que não importa quantas vezes aparece o caractere `g` , ele deve mostrar até que tenha um caractere `e` . Exemplo:

```
bigger
bege
bagggge
bugggggggge
```

7.3 Tabelas de exemplos

As tabelas a seguir mostram de uma forma mais simples os comandos explicados neste capítulo:

Expressão Regular	Descrição	Exemplo
`^`	Indica o começo da linha.	`^linux`
`$`	Indica o final da linha	`linux$`

Conjunto de caracteres e repetições

Expressão Regular	Descrição	Exemplo
`[abc]`	Conjunto de caractere único. Determina qualquer caractere dentre os pertencentes a lista.	`[Ll]inux` = Linux, linux
`[a-z]` ou `[0-9]`	Range de caractere único. Determina qualquer caractere dentre os pertencentes ao range.	`Versão[5-7]` = Versão5, Versão6, Versão7
`[^abc]`	Negação do que está na lista.	`Versão[ ^ 0-7 ]` = Versão8, Versão9
`.`	Indica qualquer caractere único.	ca.o = caso, caio, ca9o
`*`	Nenhuma ou várias ocorrências do caractere anterior.	`bola*` = bol, bola, bolaaaaa
`+`	Uma ou várias ocorrências do caractere anterior.	`bola*` = bola, bolaaaaa
`?`	Nenhuma ou apenas uma ocorrência do caractere anterior.	`bola?` = bol, bola

Conclusão

Buscar uma string específica dentro de um arquivo é uma tarefa de grande importância na manutenção de um servidor linux. Muitas vezes os arquivos de configuração do sistema são grandes, com dezenas e até centenas de linhas. Por isso, saber filtrar as nossas buscas é fundamental para um grande aumento em nossa produtividade.

A certificação LPIC foca principalmente em questões sobre busca com uma string definida e substituição de string - assuntos de que tratamos neste tópico.

No próximo capítulo, vamos trabalhar com o editor de texto VI.

CAPÍTULO 8
Editor de texto VIM

Praticamente quase toda a configuração de um sistema GNU/Linux é realizada através da edição direta de arquivos de configuração em modo texto. São diversas aplicações que podemos utilizar para editar esses arquivos, no entanto, como nosso foco é se preparar para as certificações de Linux, utilizaremos o editor VIM.

Para definirmos no Debian qual será o nosso aplicativo de texto padrão, usamos o comando:

```
# update-alternatives --config editor
Existem 5 escolhas para a alternativa editor (disponibiliza
/usr/bin/editor).

  Selecção   Caminho              Prioridade Estado
------------------------------------------------------------
* 0            /bin/nano            40        modo automático
  1            /bin/ed             -100       modo manual
  2            /bin/nano            40        modo manual
  3            /usr/bin/code        0         modo manual
  4            /usr/bin/vim.basic   30        modo manual
  5            /usr/bin/vim.tiny    15        modo manual
  6               /user/bin/vim        15        modo manual

Pressione <enter> para manter a escolha actual[*], ou digite o número da
selecção:
```

No menu que aparece, você deve digitar o número correspondente ao VIM, que no meu caso é 6, e teclar `enter` (o seu o número pode ser diferente).

8.1 Começando com o VIM

O **VIM** (VI IMproved) é uma implementação mais completa do editor **VI**. O VI foi criado por Bill Joy em 1976 para o BSD, e depois portado para diversos ambientes, como o linux. Em 1991, foi criado o VIM, estando hoje praticamente em todas as distribuições de linux.

Para instalar o VIM no debian, utilize o comando:

```
# apt-get install vim
```

Para abrir o VIM, digite:

```
# vim
```

Quando aberto, você poderá chamar a ajuda do VIM, digitando:

```
:help
```

Ou, simplesmente:

```
:h
```

A tecla é representada na maioria dos manuais e no help do VIM pelo caractere `^` (circunflexo), ou seja, o atalho para `< control > +` , aparecerá assim:

```
^L
```

O VIM trabalha com três modos de operação. Vamos supor que temos o arquivo de nome `texto.txt` . Para abrir este arquivo usando o editor VIM, usamos o comando:

```
$ vim texto.txt
```

O VIM abrirá esse arquivo no modo de **visualização**. Este modo permite que você faça a leitura do conteúdo, porém ainda não poderá escrever no arquivo.

Para que você possa editar o arquivo, é necessário entrar no modo de **inserção**. Fazemos isso pressionando a tecla `< i >` ou `< insert > .`

Quando você pressiona a tecla `< esc >`, o editor VIM entra em outro modo chamado **modo de comandos**. Nesse modo de trabalho, você poderá inserir os comandos do editor. Por exemplo, chamar a página de manual:

```
:help
```

Para acessar o modo de visualização novamente, pressione a tecla `< v >`.

8.2 Abrindo caminhos dentro de um arquivo

Caso esteja visualizando um arquivo no VIM que tenha alguma linha com referência a outro arquivo dentro do sistema, você pode abrir esta referência parando sobre a linha do arquivo, digitando `< esc >` e depois as teclas:

```
< ctrl > + < w > + < f >
```

Exemplo: considere que está lendo um arquivo e encontra no meio dele a linha:

```
O arquivo /etc/passwd contém dados dos usuário.
```

Use a tecla `< esc >`, deixe o cursor sobre a referência `/etc/passwd`, que é um arquivo de configuração de usuários no linux, e pressione `< ctrl > + < w > + < f >`. O VIM abrirá na parte superior este arquivo `/etc/passwd`. Para fechá-lo, utilize:

```
< esc >
:q
```

Abrindo arquivo em uma linha específica

Imagine a seguinte situação: você precisa encontrar a linha do usuário `Roberto` no arquivo de configuração de usuários `/etc/passwd` para alterar o Shell padrão dele de `/bin/bash` para `/bin/sh`.

Acontece que nesse cenário temos mais de 100 usuários cadastrados no sistema.

Para não ficar navegando em todo o arquivo até encontrar a linha que tem as configurações do usuário `Roberto`, podemos usar o comando `grep`, que encontra em um arquivo uma determinada palavra ou frase:

```
$ grep -n "Roberto" /etc/passwd
```

Após a execução desse comando, todas as linhas com o termo pesquisado serão exibidas no terminal:

```
$ grep -n "Roberto" /etc/passwd
27:Roberto:x:1000:1000:Juliano Ramos,,,:/home/roberto:/bin/bash
```

Caso tenha mais de uma ocorrência, elas serão exibidas assim:

```
$ grep -n "Roberto" /etc/passwd
27:Roberto:x:1000:1000:Juliano Ramos,,,:/home/roberto:/bin/bash
28:Roberto Augusto:x:1000:1000:Juliano Ramos,,,:/home/roberto2:/bin/bash
```

Agora que obtivemos o número da linha, que no meu caso foi a linha 27, podemos abrir o VIM diretamente neste ponto usando a seguinte sintaxe. Observe que o VIM iniciará já na linha 27.

```
$ vim +27 /etc/passwd
```

Alternando entre arquivos

Em certas situações é interessante abrir outros arquivos no VIM sem fechar o atual. Digamos que estou escrevendo uma documentação sobre o funcionamento de partições no linux. Em determinado momento, eu preciso verificar o arquivo que mantém as configurações de discos e partições do linux chamado `/etc/fstab`.

Para analisar esse arquivo sem que eu precise fechar o editor VIM, posso usar o recurso:

```
:e /etc/fstab
```

Nesse caso, o arquivo `fstab` será aberto na parte de cima do editor VIM (em uma janela). Eu posso alternar essa janela com o comando de teclas `<ctrl> + <6>` .

8.3 Percorrendo o texto

Estando no modo de visualização do editor VIM, o melhor modo de se movimentar pelo texto é com as teclas: **h, j, k e l**. Pode-se pensar que pelas teclas direcionais é mais fácil e, de fato, elas funcionam, mas depois de um tempo usando as letras, você perceberá que elas estão mais próximas de outras letras que são comandos.

Veja a seguir as teclas e suas funções:

```
< h > Esquerda
< j > Baixo
< k > Cima
< l > Direita
```

Agora, vamos aprender algumas outras opções, notando que são *case sensitive* (tem diferença maiúscula e minúscula).

Comando	Definição
< G >	Vai para o final do arquivo.
< gg >	Volta ao topo do arquivo.
< e >	Vai para o final de uma palavra.
< ge >	Vai para o final de uma palavra.
< w >	Vai para o começo da próxima palavra.
< b >	Vai para o começo da próxima palavra voltando ao cursor.
< f [caractere]>	Vai para o caractere definido.

Comando	Definição
< 0 >	Vai para o começo da linha.
< $ >	Vai para o final da linha.
< } >	Vai para o final do parágrafo.
< ctrl > + < G >	Mostra sua posição no arquivo.
[número da linha] + < G >	Segue para uma linha definida.

O VIM mantém um histórico de alterações. Para movimentar-se entre as alterações, use:

Comando	Definição
< g > + < ; >	Segue para a última alteração.
< g > + < , >	Segue para a próxima da lista.

Inserindo texto

Para sairmos do modo de visualização do VIM e começarmos a inserir nosso texto, usamos as seguintes teclas:

Comando	Definição
< i >	Entra no modo de inserção no local onde o cursor se encontra.
< I >	Entra no modo de inserção no começo da linha.
< a >	Entra no modo de inserção na frente de onde o cursor se encontra.
< A >	Entra no modo de inserção no final da linha.
< o >	Adiciona uma linha acima e entra em modo de inserção.

Comando	Definição
< ESC >	Volta ao modo de comandos.
< v >	Entra no modo visual.

Deletando texto

Estando no modo de visualização, podemos usar as teclas:

Comando	Definição
< x >	Deleta o caractere onde o cursor estiver.
< X >	Deleta o caractere atrás do cursor.
< d > + < w >	Deleta a partir do cursor até o começo da próxima palavra.
< d > + < $ >	Deleta o cursor até o final da linha.
< d > + < d >	Deleta a linha inteira.
< C >	Deleta de onde o cursor estiver até o final da palavra.
< c > + < e >	Deleta o cursor até o final da palavra.

Copiar e colar

Comando	Definição
< y >	Copia a palavra.

Comando	Definição
< y > + < y >	Copia a linha inteira.
< p >	Cola.

O melhor modo de se copiar algo no VIM é navegando até a linha ou palavra que se deseja, entrando no modo visual (tecla v) e selecionando a linha com as teclas de movimentação.

Após selecionar, use a tecla `< y >` para copiar, saia do modo de visualização `< esc >`, movimente-se até o local de colar e pressione `< p >`.

Recortar e colar

Para recortar e colar, use algum comando de exclusão (`x`, `dd`, `dw` ou `d$`), movimente-se até o local onde deseja colar e use a tecla `< p >`.

O último texto/palavra que você deletou será inserido após o cursor.

Substituição

Um recurso muito útil em qualquer editor de texto é a substituição de palavras. No VIM, isso é realizado no modo de comandos.

Caso esteja no modo de visualização ou inserção, pressione a tecla `< esc >` e depois `< : >` digitando algum dos comandos a seguir:

Comando	Definição
`s/antiga/nova`	Substitui a ocorrência de antiga para nova na mesma linha.
`% s/antiga/nova`	Substitui em todo o arquivo.
`% s/antiga/nova/gc`	Substitui em todo arquivo, mas solicita confirmação em cada ocorrência.

Buscando texto dentro do arquivo

Para buscar texto dentro do VIM, entramos no modo de comandos e colocamos a barra `< / >` sempre à frente dos dois pontos `< : >` e o termo a ser pesquisado, exemplos:

Comando	Definição
`/termo-a-ser-pesquisado`	Pesquisa para baixo do arquivo.
`?/termo-a-ser-pesquisado`	Pesquisa para cima do arquivo.
`//`	Busca o último termo pesquisado.

Ao entrar no modo de busca, o VIM deixa a palavra pesquisada na barra inferior, então podemos usar:

Comando	Definição
`< n >`	Para a próxima ocorrência.
`< N >`	Para a ocorrência anterior.

Conclusão

Praticamente todos os servidores que rodam sobre o linux utilizam arquivos de configuração em texto puro. Como não usamos interface gráfica em um servidor linux por questão de desempenho e simplicidade ao ambiente, conhecer o VIM é fundamental para a administração do sistema. O editor VIM é cobrado no tópico 103.8 da certificação LPIC-1.

CAPÍTULO 9
Arquitetura do sistema

Neste tópico, abordaremos como detectar e analisar dispositivos de hardware. O capítulo é direto ao ponto. O conteúdo é 100% criado com base nas questões do exame 101 sobre o tema.

9.1 Introdução básica ao hardware

Em um computador, um pedido de interrupção, do inglês *interrupt request (IRQ)*, é um sinal de hardware enviado ao processador que temporariamente pausa um programa em execução e permite que um programa especial, um manipulador de execução, seja executado.

Interrupções de hardware são usadas para manipular eventos como o recebimento de dados de uma placa de rede e/ou pressionamentos de tecla do teclado ou simplesmente movimentos que você realiza com o mouse.

O BIOS (um acrônimo de *Basic Input/Output System*, em português Sistema Básico de Entrada/Saída) é o firmware não volátil usado para realizar a inicialização do hardware durante o processo de boot do seu computador. Ele vem pré-instalado na memória permanente da placa mãe do computador e é o primeiro software a ser executado quando você liga a máquina.

Para a certificação Linux LPIC é importante você saber os endereços mais comuns de IRQ bem como conhecer o arquivo que mantém essas informações.

IRQ	Definição

IRQ	Definição
IRQ 0	Sinal de clock da placa mãe.
IRQ 1	Teclado.
IRQ 2	Cascateador de IRQs.
IRQ 3	Porta Serial 2 (RS-232).
IRQ 4	Porta Serial 1 (RS-232).
IRQ 5	Livre.
IRQ 6	Drive de disquete.
IRQ 7	Porta paralela (impressoras).
IRQ 8	Relógio do CMOS.
IRQ 9	Placa de vídeo.
IRQ 10	Livre.
IRQ 11	Controlador USB.
IRQ 12	Porta PS/2.
IRQ 13	Coprocessador aritmético.
IRQ 14	IDE primária.
IRQ 15	IDE Secundária.

O arquivo do linux que contém informações relacionadas ao IRQ é o arquivo `/proc/interrupts` .

Endereços de I/O - Entrada e saída

As portas de entrada e saída são endereços de memória reservados no microprocessador para os dispositivos realizarem trocas de informações de forma simplificada.

Os dispositivos podem usar mais de uma porta de entrada/saída ou uma faixa de endereços. Por exemplo, uma placa de som padrão usa as portas 0x220, 0x330 e 0x388. Mas cada dispositivo possui um endereço de porta única que não pode ser compartilhado entre outros dispositivos.

Os endereços de entrada e saída, E/S, ou I/O, do inglês *Input/output*, em uso no sistema podem ser visualizados no arquivo `/etc/ioports` .

Endereços DMA

A DMA (*Direct Memory Access*, em português Acesso Direto a Memória) é usada para permitir a transferência de dados entre dispositivos e a memória sem a intervenção do processador.

Esse acesso é feito através de canais (*channels*). A maioria dos computadores tem dois controladores de DMA. O primeiro controla os canais 0, 1, 2 e 3, e o segundo, os canais 4, 5, 6 e 7, totalizando 8 canais.

Os canais de DMA em uso no sistema podem ser visualizados com o comando:

```
# cat /proc/dma
4: cascade
```

Alguns dispositivos como portas seriais e paralelas já utilizam os recursos no padrão a seguir:

TABELA – Referência de dispositivos no Linux

Nome do dispositivo no Linux	Nome do dispositivo no Windows	Porta E/S	DMA	IRQ
ttys0	COM1	0x3F8	-	4
ttys1	COM2	0x2F8	-	3

Nome do dispositivo no Linux	Nome do dispositivo no Windows	Porta E/S	DMA	IRQ
ttys2	COM3	0x3E8	-	4
ttys3	COM4	0x2E*	-	3
lp0	LPT1	0x378	-	7
lp1	LPT1	0x278	-	5
/dev/sda1	C:	0x1F0	-	14

Barramentos

Barramento é a interface de hardware na qual você conecta seus dispositivos de forma física na placa mãe. Os principais barramentos que são cobrados no exame LPIC são: PCI e USB.

Para visualizarmos os dispositivos que estão conectados em nosso barramento PCI, usamos o comando:

```
# lspci
00:00.0 Host bridge: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 30h-3fh) Processor Root Complex
00:01.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Kaveri [Radeon R7 Graphics] (rev d6)
.....
```

A saída do comando pode ser extensa dependendo da quantidade de dispositivos que você possui em seu computador. Na saída do meu comando, observe que o dispositivo *VGA compatible controller: Advanced Micro Devices* está localizado no endereço 00:01.1. Podemos obter uma informação mais detalhada sobre esse dispositivo utilizando o comando:

```
# lspci -s 00:01 -v
00:01.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Kaveri [Radeon R7 Graphics] (rev d6) (prog-if 00 [VGA controller])
```

```
    Subsystem: ASRock Incorporation Kaveri [Radeon R7 Graphics]
    Flags: bus master, fast devsel, latency 0, IRQ 29
    Memory at c0000000 (64-bit, prefetchable) [size=256M]
    Memory at d0000000 (64-bit, prefetchable) [size=8M]
    I/O ports at f000 [size=256]
    Memory at feb00000 (32-bit, non-prefetchable) [size=256K]
    Expansion ROM at 000c0000 [disabled] [size=128K]
    Capabilities: <access denied>
    kernel driver in use: radeon
    kernel modules: radeon, amdgpu
```

Para visualizar os dispositivos conectados ao nosso barramento USB, usamos o comando:

```
# lsusb
Bus 003 Device 002: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 003 Device 001: ID 1d6b:0002 linux Foundation 2.0 root hub
Bus 007 Device 002: ID 275d:0ba6  USB OPTICAL MOUSE
Bus 007 Device 001: ID 1d6b:0001 linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0002 linux Foundation 2.0 root hub
Bus 005 Device 007: ID 1a2c:0c21 China Resource Semico Co., Ltd USB
Keyboard
Bus 005 Device 001: ID 1d6b:0001 linux Foundation 1.1 root hub
Bus 001 Device 002: ID 090c:1000 Silicon Motion, Inc. - Taiwan (formerly
Feiya Technology Corp.) Flash Drive
Bus 001 Device 001: ID 1d6b:0002 linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 linux Foundation 1.1 root hub
```

Para obter detalhes sobre um dispositivo específico, por exemplo, meu mouse (USB OPTICAL MOUSE) que está localizado no BUS 007 e DEVICE 002, usamos o comando:

```
# lsusb -s 007:002 -v

Bus 007 Device 002: ID 275d:0ba6  USB OPTICAL MOUSE
Couldn't open device, some information will be missing
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.10
```

```
  bDeviceClass            0
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x275d
  idProduct          0x0ba6
  bcdDevice            1.00
...
```

9.2 As partições virtuais

Partições virtuais do Linux são partições dinâmicas que contêm informações de tudo o que ocorre no sistema operacional em tempo real. São alocadas na memória RAM e são uma ótima fonte para se obter log do sistema. São elas:

Partição virtual	Definição
`/proc`	Contém informações dos processos ativos e recursos de hardware.
`/sys`	Contém informações sobre dispositivos de hardware.
`/dev`	Referências aos dispositivos de sistema, inclusive armazenamento.

Em um sistema operacional linux, tudo é um arquivo ou um processo. Um dispositivo de armazenamento apontado como `/dev/sda1` é por sua vez o caminho ao acesso físico, ou seja, o próprio dispositivo. O linux utiliza um software chamado `udev`, que gerencia os dispositivos de armazenamento localizados em `/dev`.

Outra aplicação importante do kernel linux é o `dbus/hald`, que é responsável pela comunicação entre os processos do linux. Entre

outras funções, ele informa quando um novo hardware é adicionado e é ele quem espalha essas informações aos processos.

Você pode visualizar as partições do seu sistema operacional e as partições virtuais montadas em sua memória RAM através do comando:

```
# df -a
```

No diretório virtual `/proc` , temos alguns arquivos que são importantes na administração de um servidor linux. São eles:

Arquivo	Definição
`/proc/interrupts`	Informação de IRQ.
`/proc/dma`	Informação de DMA.
`/proc/ioports`	Informação de entrada e saída.
`/proc/meminfo`	Informação da memória RAM.
`/proc/cpuinfo`	Informação sobre a CPU.

Arquitetura do kernel linux

Ao instalar o debian ou outra distribuição linux, estamos utilizando o kernel que foi compilado por seus desenvolvedores. Este kernel padrão deve ser capaz de ser executado em praticamente qualquer PC e de dar suporte a diversos tipos de recursos que o usuário pretenda utilizar, como sistemas de arquivos, usb, diferentes tipos de protocolos de rede, entre outros.

A pessoa desenvolvedora compila um kernel básico, com poucas funcionalidades, e cria pedaços de código com funções específicas que podem ser adicionados a ele. Esses pedaços de código são chamados de módulos.

Quando o sistema é carregado, esse kernel básico é colocado na memória e passa a controlar a máquina. Digamos que você possui

uma partição com o sistema de arquivos NTFS (Sistema de arquivos do Windows 10) que não está no kernel básico padrão. Durante a inicialização do sistema, o kernel básico vai procurar se o módulo que dá suporte para esse tipo de sistema de arquivos está disponível. Se esse módulo for encontrado, ele será carregado, expandindo as funcionalidades do kernel.

9.3 Trabalhando com módulos

O desenvolvimento de uma nova funcionalidade para o kernel linux pode ser feito diretamente no kernel ou através de uma compilação como um módulo. De modo geral, a escolha tende a ser a compilação como módulo, o que significa que o kernel só vai ativar o módulo caso seja necessário. Essa opção otimiza o sistema e torna o kernel do linux muito dinâmico.

Vamos a um exemplo prático:

Digamos que você esteja usando uma impressora USB. Neste momento, o módulo para ela funcionar está ativo no sistema. Depois de um tempo, você para de usar essa impressora por qualquer motivo e, nesse caso, o módulo de suporte a impressora não é mais necessário e você pode desabilitá-lo.

Ao executar o comando `lsmod`, você pode visualizar quais módulos estão carregados atualmente na memória:

```
# lsmod
Module                  Size  Used by
fuse                   98304  3
pci_stub               16384  1
vboxpci                24576  0
vboxnetadp             28672  0
vboxnetflt             28672  0
vboxdrv               454656  3 vboxnetadp,vboxnetflt,vboxpci
xfrm_user              36864  2
```

```
xfrm4_tunnel           16384  0
tunnel4                16384  1 xfrm4_tunnel
ipcomp                 16384  0
xfrm_ipcomp            16384  1 ipcomp
esp4                   20480  0
ah4                    20480  0
af_key                 36864  0
xfrm_algo              16384  5 xfrm_user,esp4,ah4,af_key,xfrm_ipcomp
snd_usb_audio         180224  2
uvcvideo               90112  0
snd_usbmidi_lib        28672  1 snd_usb_audio
snd_rawmidi            32768  1 snd_usbmidi_lib
videobuf2_vmalloc      16384  1 uvcvideo
videobuf2_memops       16384  1 videobuf2_vmalloc
videobuf2_v4l2         24576  1 uvcvideo
videobuf2_core         40960  2 uvcvideo,videobuf2_v4l2
wl                   6365184  0
videodev              176128  3 uvcvideo,videobuf2_core,videobuf2_v4l2
media                  40960  2 uvcvideo,videodev
snd_seq_device         16384  1 snd_rawmidi
arc4                   16384  2
```

Cortei a saída acima porque o meu kernel tem muitos módulos ativos. Eles são diferentes de máquina para máquina, porque possuímos hardware e configurações diferentes. Durante a instalação do debian, o sistema verifica seu equipamento e define quais módulos vão estar ou não ativos no seu sistema.

Os módulos possuem dependências e você pode saber a quantidade de dependências por módulos na coluna `used` na saída do comando `lsmod`. A coluna `size` mostra o tamanho do módulo em bytes. Um módulo só pode ser removido se o número de dependências for zero.

Os módulos são carregados através do comando `insmod`. Se o módulo B depende do A, este deve ser carregado antes. Digamos que na coluna `used` do módulo A agora esteja mostrando 1, referente ao módulo B que você carregou. Não será possível

remover o módulo A se você não remover sua dependência B primeiro.

A seguir, vamos observar quais são os procedimentos para carregar o módulo `vfat` que dá suporte ao sistema de arquivos FAT32, muito comum em pendrivers formatados no Microsoft Windows. Depois veremos como é o procedimento para sua remoção.

O primeiro passo é verificar se o módulo em questão já não está ativo no sistema. Para isso, executamos:

```
# lsmod | grep vfat
```

Caso não obtenha nenhum retorno, significa que o seu módulo não está ativo. Se tiver um retorno como:

```
vfat                   20480  0
fat                    69632  1 vfat
```

Significa que o seu módulo está ativo e funcionando. Para carregar um módulo, usamos a sintaxe:

```
# modprobe nome_do_modulo
```

Nesse caso:

```
# modprobe vfat
```

Para remover o módulo:

```
# rmmod vfat
```

Para saber informações sobre um módulo e ver suas dependências, utilize o comando `modinfo` . Por exemplo:

```
# modinfo vfat
filename:       /lib/modules/4.9.0-deepin9-amd64/kernel/fs/fat/vfat.ko
author:         Gordon Chaffee
description:    VFAT filesystem support
license:        GPL
alias:          fs-vfat
depends:        fat
```

```
intree:         Y
vermagic:       4.9.0-deepin9-amd64 SMP preempt mod_unload modversions
```

Observe na saída que a dependência para o `vfat` chama-se `fat`. Ao ativar o `vfat`, vamos obter novamente informação sobre ele:

```
# lsmod | grep vfat
vfat                   20480  0
fat                    69632  1 vfat
```

Observe que `fat` tem na saída da coluna `used` o número 1. Vamos tentar removê-lo:

```
# rmmod fat
rmmod: ERROR: Module fat is in use by: vfat
```

Veja que foi impossível remover, já que ele está sendo usado por `vfat`. Então vamos remover o `vfat` primeiro:

```
# rmmod vfat
```

O modo `vfat` foi removido com sucesso, mas ainda temos a dependência `fat`, que precisa ser removida:

```
# rmmod fat
```

Para remover de modo automático um módulo e suas dependências, podemos usar o comando `modprobe -r`. Veja:

```
# modprobe -r vfat
```

O bom nesse caso é que, além de remover o `vfat`, ele remove o `fat`, que é sua dependência.

9.4 Quais módulos estão disponíveis?

Para saber quais módulos estão disponíveis no seu sistema, acesse o diretório `/lib/modules/versao-do-kernel/kernel`. No meu caso:

```
# cd /lib/modules/4.9.0-debian-amd64/kernel/
```

O meu kernel chama-se `4.9.0-debian-amd64` , o seu provavelmente terá um nome diferente já que esse kernel que estou utilizando não é o padrão do debian. Dentro desse diretório, você vai encontrar alguns subdiretórios como:

```
arch  crypto  drivers  fs  lib  mm  net  security  sound  virt
```

Dentro de cada um deles você encontra os módulos. Observe o diretório `fs - file system` , que é o diretório que contém módulos sobre sistema de arquivos. Dentro dele, você pode visualizar todos os diretórios dos módulos disponíveis de `fs` :

```
# cd /lib/modules/4.9.0-debian-amd64/kernel/fs
# ls
9p       befs            ceph      dlm       ext4      fuse     jbd2
minix       nilfs2  overlayfs  reiserfs  udf
adfs     bfs             cifs      ecryptfs  f2fs      gfs2     jffs2
ncpfs       nls     pstore     romfs     ufs
affs     binfmt_misc.ko  coda      efivarfs  fat       hfs      jfs
nfs         ntfs    qnx4       squashfs  xfs
afs      btrfs           configfs  efs       freevxfs  hfsplus  lockd
nfs_common  ocfs2   qnx6       sysv
autofs4  cachefiles      crypto    exofs     fscache   isofs    mbcache.ko
nfsd        omfs    quota      ubifs
```

No meu caso, eu tenho diversos módulos de sistemas de arquivo. Dentro do diretório do módulo de sua escolha, você obtém o nome do módulo, por exemplo:

```
# cd /lib/modules/4.9.0-debian-amd64/kernel/fs/fat
# ls
# fat.ko  msdos.ko  vfat.ko
```

Observe que dentro do diretório `/lib/modules/4.9.0-debian-amd64/kernel/fs/fat/` obtive o nome dos módulos disponíveis: `vfat, msdos, fat .`

Conclusão

A arquitetura de sistemas linux é um assunto abrangente e, em alguns casos, complexo. Porém, a certificação LPI exige apenas o conhecimento relativo aos assuntos tratados neste capítulo, como a verificação de um módulo ativo no sistema. Verifique, como laboratório, quais módulos seu sistema operacional mantém ativo, ative alguns módulos complementares e depois faça sua remoção.

Capítulo 10
O boot do sistema

Basicamente o processo de boot do Linux acontece na seguinte ordem:

1. BIOS
2. MBR
3. Bootloader (GRUB2)
4. initrd/initramfs

BIOS, cuja definição já vimos no capítulo anterior, trata-se de um mecanismo responsável por algumas atividades consideradas corriqueiras em um computador, mas que são de suma importância para o correto funcionamento de uma máquina. Se a BIOS para de funcionar, o PC também para.

MBR é a sigla para *Master Boot Record*. Hoje, MBR se encontra em processo de substituição pelo GPT, que significa *GUID Partition Table*. Entre as limitações do MBR está o fato de não permitir ao usuário estabelecer mais do que quatro partições primárias no mesmo disco. Por outro lado, o GPT pode criar até 128 partições diferentes dentro de um mesmo disco.

O boot loader é o aplicativo de gestão de arranque de boot do linux e fica instalado na MBR ou em uma partição especial quando o sistema é GPT.

O gerenciador de boot tem a função de carregar o sistema de inicialização na memória e, depois que o sistema começar a inicializar, transferir essas informações para o disco. É no software gerenciador de boot que ficam as informações dos sistemas operacionais que você possui em seu computador. Se você tem Windows e Linux (dual boot), quando inicia o computador, o gerenciador de boot oferece um menu de escolha para você decidir qual sistema deseja iniciar.

Atualmente, o boot loader padrão é o GRUB2 (Grand Unified Bootloader).

O `initrd/initramfs` é um filesystem temporário que carrega informações essenciais na memória RAM, como módulos para alguns dispositivos. Esses módulos podem ser drivers para que determinado hardware funcione, ou alguma funcionalidade, como o suporte a sistema de arquivo NTFS.

O `initrd/initramfs` gera um processo que chamamos de `init` . Esse processo sempre recebe o PID (identificação do processo) 1.

Uma forma de você visualizar o log gerado no processo de boot é usando o comando `dmesg` :

```
# dmesg
```

10.1 Alterando os runlevels, Shutdown e Reboot

A LPI menciona que devemos conhecer o sistema de inicialização do linux. Este sistema recebe atualmente o nome de `systemd` . Systemd (*System and Service Manager* ou, em português, Gerenciador de sistemas e serviços) é o software responsável pelo sistema `init` que gerencia os processos e o sistema de inicialização do linux (boot).

Comandos do systemd

Para saber se você utiliza de fato o `systemd` , use o comando:

```
# ls -l /sbin/init
```

Se o resultado do comando for:

```
/sbin/init -> /lib/systemd/systemd
```

Significa que o sistema que você usa é de fato o `systemd`. Caso não tenha essa saída, você está usando o `systemv`.

Para listar as units, usamos o comando:

```
# systemctl list-units
```

As unidades são as configurações que estão em `/lib/systemd/system`. Para reiniciar a máquina ou desligar, usamos:

```
# systemctl reboot
# systemctl poweroff
```

Também podemos definir o `target` padrão. Para ver o `target` atual (semelhante ao `runlevel`), usamos:

```
# systemctl get-default
```

Para definir um `target`, usamos:

```
# systemctl set-default multi-user.target
```

Para alterar somente nesta sessão, usamos:

```
# systemctl isolate rescue.target
```

`rescue.target` seria semelhante ao **init 1** do `systemv`. Para iniciar o `target` padrão definido no sistema:

```
# systemctl default
```

O `target` gráfico do `systemd` é o **graphical.target**.

Iniciar, parar e reiniciar serviços

Para obter informação sobre o estado de um serviço, usamos:

```
# systemctl status cron.service
```

Para iniciar um serviço, usamos:

```
# systemctl start cron.service
```

Para reiniciar um serviço, usamos:

```
# systemctl restart cron.service
```

Para parar um serviço, usamos:

```
# systemctl stop cron.service
```

Comandos de shutdown e restart

Temos dois runlevels específicos para desligar e reiniciar. Temos o **runlevel 0**, que desliga, e o **runlevel 6**, que reinicia. Mas existe uma forma melhor de fazer isso, usando o comando `shutdown`.

Se você usar o comando `shutdown` sem nenhum parâmetro, o sistema será desligado em 1 minuto, mas é possível cancelar usando o comando `shutdown -c`. Por exemplo:

```
# shutdown
# shutdown -c
```

Para reiniciar:

```
# shutdown -r
```

Para desligar o linux e não desligar a máquina (ou seja, a fonte de energia), usamos:

```
# shutdown -h
```

Para agendar o horário de desligamento:

```
# shutdown 18:00
```

Para agendar o desligamento para daqui a um certo tempo (minutos):

```
# shutdown -r +10
# shutdown -h +10
```

Também temos os comandos `reboot` e `poweroff`, usados respectivamente para reiniciar e desligar o sistema.

Conclusão

O exame cobra muito o seu conhecimento sobre `systemd`, que é o sistema de inicialização atual do mercado. Realize testes de laboratório, alterando como seu sistema inicializa e colocando modo texto e depois modo gráfico. Inicie, veja o status e pare serviços. Sobre o `systemv`, apenas memorize que o arquivo de configuração está em `/etc/inittab` e que visualizamos os modos de inicialização com o comando `runlevel`.

Capítulo 11
Particionamento de discos

Além da parte de comandos, a prova da certificação LPI aborda conceitos. Este capítulo demonstra como podemos realizar um sistema de particionamento de discos seguindo os padrões estabelecidos no exame.

As vantagens em particionar um disco são:

1. Melhor gestão do espaço do disco.
2. Diferentes tipos de *filesystem* para cada partição.
3. Proteção contra erros do disco.
4. Diferentes níveis de segurança.
5. Backup facilitado.

11.1 Sistemas de particionamento

Basicamente temos dois modos de trabalhar com particionamento de discos: o modo MBR e o modo GPT. MBR e GPT são padrões que determinam a forma como os dados são armazenados no disco. Isso inclui a forma como as partições são distribuídas e como esses drivers de SSDs ou HDs se comportam.

Revisando, MBR é a sigla para *Master Boot Record* e GPT significa *GUID Partition Table*. Enquanto o modelo MBR permite a criação de apenas 4 partições primárias, o modelo GPT permite 128 partições. Outra diferença é que o modelo MBR permite apenas partições com 2TB, já o GPT permite 9.4ZB (Zetabytes).

É possível instalar o linux em apenas uma partição, chamada `root` (raiz). Não confundir com o nome do usuário administrador do linux.

No entanto, recomendamos pelo menos duas partições, sendo a `root` e a partição `swap`.

A área de SWAP é um espaço no disco que simula a memória RAM e é utilizada quando o espaço total da RAM é usado. Outra maneira de organizar o nosso disco é usando o LVM - `Logical Volume Manager`, que abordaremos em seguida. Sobre a SWAP, é possível a usarmos em uma partição definida no disco, ou em um arquivo. Vejamos como podemos criar a partição `swap`.

Criaremos um arquivo em `/mnt` chamado `mem.swap`:

```
sudo dd if=/dev/zero of=/mnt/mem.swap bs=1M count=512
```

Daí seguiremos com a criação do sistema de arquivos no arquivo alvo:

```
sudo mkswap /mnt/mem.swap
```

E ativamos a nova SWAP:

```
 sudo swapon /mnt/mem.swap
```

No detalhe acima, os 512MB já foram somados aos que já existiam, então para tornar as mudanças definitivas (para que essa SWAP esteja disponível para uso do sistema a cada reinicialização) adicionaremos uma entrada no arquivo `/etc/fstab` assim:

```
sudo nano /etc/fstab
/mnt/mem.swap swap swap defaults 0 0
```

Feito isso, basta reiniciar o sistema e testar com o comando:

```
# free -h
```

11.2 Particionamento dos discos

Primeiramente, vamos verificar as partições existentes em nosso sistema. Além do disco rígido onde o sistema operacional está instalado, acrescente mais um disco rígido de 20 GB para testes. Vamos digitar no terminal o comando a seguir como `root` :

```
fdisk -l
```

```
Dispositivo Boot      Início        Fim      Blocos   Id  Sistema
/dev/sda1   *       2048    34125823    17061888   83  Linux
/dev/sda2       34127870    37746687     1809409    5  Estendida
/dev/sda5       34127872    37746687     1809408   82  Linux swap / Solaris
```

Figura 11.1: Partições do disco.

Observe que, além das informações sobre meus particionamentos em `/dev/sda` , ele mostra o disco `/dev/sdb` :

```
Disco /dev/sdb: 21.5 GB, 21474836480 bytes
255 cabeças, 63 setores/trilhas, 2610 cilindros, total de 41943040 setores
Unidades = setores de 1 * 512 = 512 bytes
Tamanho do setor (lógico/físico): 512 bytes / 512 bytes
Tamanho da E/S (mínimo/ideal): 512 bytes / 512 bytes
Identificador do disco: 0x00000000

O disco /dev/sdb não contém uma tabela de partições válida
```

Figura 11.2: Disco /dev/sdb.

Podemos verificar que existem três partições no disco `sda` :

- **/dev/sda1** – Marcada como boot (inicializável), é onde está instalado nosso linux. Essa partição, em nosso caso, se inicia no setor 2048 do disco e termina no setor 34125823, possuindo 17061888 blocos. Veja que a partição possui um Id, que é um código mostrando o tipo de partição criada, no caso, “Linux”.
- **/dev/sda2** – Inicia no bloco 34127870 e termina no bloco 37746687, e possui Id 5, que significa que se trata de uma partição estendida do disco (contêiner para partições lógicas).
- **/dev/sda5** – Inicia no bloco 34127872 e termina no no bloco 3774668 (mesmo que o sda2), e possui Id 82, que corresponde

a uma partição de `swap`. O número 5 no sda5 também nos indica que se trata de uma unidade lógica.

Agora, veja que para o disco `/dev/sdb` não temos essas informações sobre as partições, pois ele acabou de ser adicionado e, portanto, ainda não foi particionado. Vamos criar então duas partições nesse disco. Execute o comando a seguir para entrarmos no `fdisk` e configurar o disco de forma interativa:

```
fdisk /dev/sdb
```

Digite `m` no prompt interativo para ver as opções de comandos do `fdisk`:

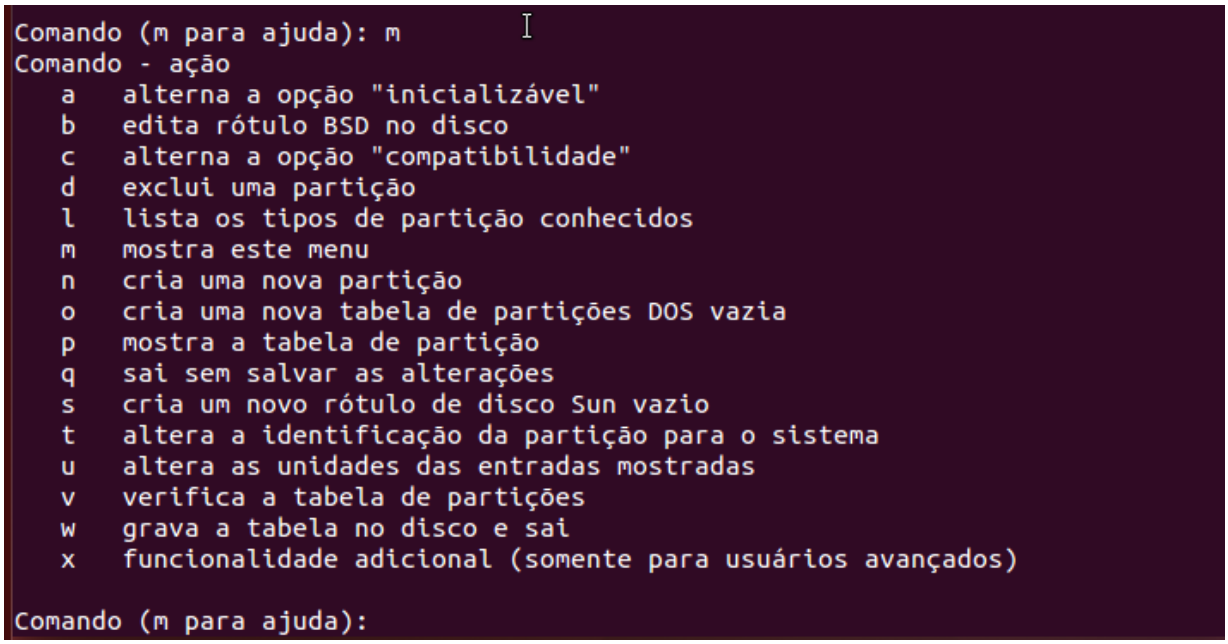
Comando (m para ajuda): m
Comando - ação
a alterna a opção "inicializável"
b edita rótulo BSD no disco
c alterna a opção "compatibilidade"
d exclui uma partição
l lista os tipos de partição conhecidos
m mostra este menu
n cria uma nova partição
o cria uma nova tabela de partições DOS vazia
p mostra a tabela de partição
q sai sem salvar as alterações
s cria um novo rótulo de disco Sun vazio
t altera a identificação da partição para o sistema
u altera as unidades das entradas mostradas
v verifica a tabela de partições
w grava a tabela no disco e sai
x funcionalidade adicional (somente para usuários avançados)
Comando (m para ajuda):

Figura 11.3: Opções do particionador.

Veja as opções disponíveis. Por exemplo, o comando `d` exclui uma partição, o `n` cria uma nova partição e o `t` altera o tipo da partição criada.

Vamos criar uma nova partição no disco. Pressione a letra `n` e depois tecle `enter`.

Será perguntado o tipo de partição desejada. Podemos escolher entre primária (letra `p`) ou estendida (letra `e`). Caso já tenhamos uma partição estendida, teremos também a opção de criar unidade lógica. Digite `p` para criarmos uma partição primária no disco. Note que essa é a opção padrão, ou seja, se você apenas pressionar `enter` sem digitar nada, será assumida essa opção como escolha:

```
Select (default p): p
Número da partição (1-4, padrão 1):
```

Figura 11.4: Escolha do número da partição.

Agora vamos escolher o número da partição, de 1 a 4 (podemos ter até quatro partições primárias, ou três primárias e uma estendida, nesse esquema tradicional de particionamento.

Vamos agora delimitar nossa partição e determinar seu tamanho. Digite o número do primeiro setor (onde a partição se inicia). Por padrão, o setor será o de número 2048, mas você pode iniciar a partição em qualquer setor, desde que acima desse número. Tecle `enter` para aceitar o padrão.

```
Primeiro setor (2048-41943039, padrão 2048):
Usando valor padrão 2048
Último setor, +setores ou +tamanho{K,M,G} (2048-41943039, padrão 41943039):
```

Figura 11.5: Tamanho da partição.

A seguir, vamos determinar onde termina a partição entrando com o valor do último setor. Podemos também entrar com o número de setores desejados (cada um tem 512 bytes) ou ainda diretamente com o tamanho desejado, digitando `+NX`, onde N é o tamanho desejado e X é uma letra indicando a unidade de armazenamento que pode ser K para Kilobytes, M para Megabytes e G para Gigabytes.

Por exemplo, vamos criar uma partição de 5GB. Para isso, digite `+5G` e pressione `enter`.

Pronto, a partição foi dimensionada com sucesso (não apareceram mensagens de erro!). Podemos verificar digitando o comando `p` e pressionando `enter`:

```
Comando (m para ajuda): p

Disco /dev/sdb: 21.5 GB, 21474836480 bytes
255 cabeças, 63 setores/trilhas, 2610 cilindros, total de 41943040 setores
Unidades = setores de 1 * 512 = 512 bytes
Tamanho do setor (lógico/físico): 512 bytes / 512 bytes
Tamanho da E/S (mínimo/ideal): 512 bytes / 512 bytes
Identificador do disco: 0x070ebcd0

Dispositivo Boot      Início         Fim      Blocos   Id  Sistema
/dev/sdb1            2048    10487807     5242880   83  Linux

Comando (m para ajuda):
```

Figura 11.6: Exibir as partições.

Quando você criar as partições desejadas, salve o particionador com a tecla `w`.

LVM - Logical Volume Manager

LVM (*Logical Volume Manager*) é um método de alocar espaço do disco rígido em volumes lógicos que podem ser facilmente redimensionados, ao contrário das partições. Com o LVM você poderá aumentar e/ou diminuir o tamanho das suas partições.

Para este nosso laboratório, adicione um disco sem conteúdo em sua máquina virtual ou física. Ele será reconhecido no linux como `/dev/sdb`. Crie três partições neste disco.

- /dev/sdb1
- /dev/sdb2
- /dev/sdb3

Agora que possuímos as partições, adicione-as ao esquema LVM com o comando:

```
pvcreate /dev/sdb1
pvcreate /dev/sdb2
pvcreate /dev/sdb3
```

Visualize as partições criadas com o comando:

```
pvdisplay
```

O próximo passo é unirmos essas partições para criar um único volume de dados. Neste caso, execute:

```
vgcreate storage /dev/sdb1 /dev/sdb2
```

Por enquanto, não usaremos a partição `/dev/sdb3` para aprendermos como adicioná-la ao volume de nome (storage) mais à frente.

Para verificar informações sobre esse volume de dados que possui as partições `/dev/sdb1` e `/dev/sdb2`, use o comando:

```
vgdisplay -v storage
```

O próximo passo é criarmos o volume lógico, necessário para criarmos o espaço a ser usado para armazenamento de dados.

```
lvcreate -L 3G -n lv_storage storage
```

Podemos visualizar este espaço criado com o nome de `lv_storage` a partir do grupo `storage`, com o comando:

```
lvdisplay
```

Agora que está tudo pronto, vamos adicionar a partição `/dev/sdb3` a esse esquema de particionamento. Para isso, executamos:

```
vgextend storage /dev/sdb3
```

Para usarmos esse espaço que é a soma das partições `/dev/sdb1`, `/dev/sdb2` e `/dev/sdb3`, temos que formatar esta partição:

```
mkfs.ext4 /dev/storage/lv_storage
```

Em seguida, montar o acesso:

```
mount -t ext4 /dev/storage/lv_storage /mnt/storage
```

Após usar esse espaço para armazenamento de dados, você observou que ele está pequeno, então você deseja adicionar mais espaço. Para isso, é possível usar o espaço adicional que está no seu grupo de volume de nome (`storage`).

O primeiro passo é desmontar a conexão:

```
umount /dev/storage
```

Adicione o espaço desejado:

```
lvextend -L +2G/dev/storage/lv_storage
```

Realize verificação do disco para que ele entenda o novo tamanho:

```
e2fsck -f /dev/storage/lv_storage
resize2fs /dev/storage/lv_storage
```

Monte novamente a partição:

```
mount -t ext4 /dev/storage/lv_storage /mnt/storage
```

11.3 Comandos essenciais

O comando `df -h` é o responsável pela visualização dos pontos de montagem, assim como mostra o espaço livre nestes locais:

```
df -h
```

A partição `swap` não aparece no comando `df -h` , como você observará. Para visualizá-la, utilizamos outro comando:

```
cat /proc/swaps
```

Agora, se você deseja visualizar as partições sem que o comando retorne os pontos de montagem, utilize o comando a seguir:

```
fdisk -l
```

Conclusão

Um trabalho comum do administrador de sistemas é a instalação e configuração de servidores linux. Para essa, tarefa conhecer sobre particionamento de disco é algo essencial. A certificação linux cobra do aluno um bom conhecimento sobre como criar partições e realizar o seu acesso, conteúdo que abordamos aqui neste capítulo. No exame atual, você terá uma média de sete perguntas sobre este tema.

Capítulo 12
Configuração de boot loader

O *boot loader* (gestor de boot) é responsável pela escolha do kernel linux durante a inicialização do sistema, assim como a escolha de outro sistema, por exemplo, o Windows. A certificação exige um conhecimento superficial sobre ele. Por isso, vamos abordar os principais arquivos de configuração. Sem um *boot loader*, você não teria como realizar *dual boot* em seu computador, porque é ele que oferece o menu para a escolha de qual versão do kernel linux ou de outro sistema você deseja iniciar.

Principais arquivos de configuração

`/boot/grub/grub.cfg` - Principal arquivo de configuração. É um grande script somente leitura com informações da configuração do grub2.

`/etc/grub.d/` - Diretório que contém diversos scripts usados para a construção do arquivo de configuração `grub.cfg` .

`/etc/default/grub` - Arquivo de configuração do menu do GRUB2, lido pelos scripts do GRUB2 e então escrito no `grub.cfg` .

Criando uma entrada para o Windows

Dentro do arquivo de configuração `/etc/grub.d` temos o arquivo `40_custom` que é usado para criar entradas para outros sistemas operacionais. O primeiro passo é dar permissão de execução para este arquivo:

```
chmod +x 40_custom
```

Em seguida, criamos a entrada. No exemplo a seguir, vamos adicionar uma entrada para o Windows que está em um segundo HD (sdb). O resultado seria:

```
#!/bin/sh
exec tail -n +3 $0

menuentry "Windows 7" --class windows --class os {
  insmod part_msdos
  insmod ntfs
  set root='(/dev/sdb,msdos1)'
  chainloader +1
}
```

Trocar a senha de root - Grub

Na tela de login tecle `< e >`, em seguida edite a linha deixando-a como no exemplo a seguir:

```
linux /vmlinuz-2.6.32-5-686 root=UUID=1c5f16d2-b410-403b-a7a9-
382e3f\1071b6 ro init=/bin/bash
```

Tecle `< ctrl > + < x >` para iniciar o sistema. Quando chegar no Shell, digite:

```
mount -o remount,rw /
passwd root
```

12.1 Bibliotecas compartilhadas

A prova contém apenas uma questão sobre este tópico (peso 1), sendo assim é um tópico rápido de ser abordado. A biblioteca compartilhada é um pedaço de código que está fora de sua aplicação, de modo que facilite para a pessoa desenvolvedora a criação de um software. Muitas vezes diversos recursos são compartilhados entre os programas, por exemplo:

Ao desenvolver uma aplicação que rode um tipo de vídeo, eu não preciso desenvolver o codec para aquele formato de vídeo, eu posso apenas chamar esse codec dentro do código do meu programa, reduzindo o tempo de desenvolvimento da aplicação.

A biblioteca estática é incorporada na minha aplicação. A desvantagem é que o programa vai ficar maior, porque, quando eu compilar minha aplicação, todas as bibliotecas necessárias estarão dentro do código.

O modo mais inteligente é a biblioteca dinâmica, que faz apenas uma referência no código, a biblioteca vai ser carregada apenas uma vez e serve para diversos aplicativos.

Comandos básicos

Para saber quais são as bibliotecas que uma aplicação está usando, o primeiro passo é descobrir o caminho do executável:

```
whereis vim.basic
```

Agora que você sabe o caminho do executável, execute o comando:

```
ldd /usr/bin/vim.basic
```

Tudo que tem `so` são bibliotecas, veja:

```
linux-vdso.so.1 =>  (0x00007ffdfeb23000)
    libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5
(0x00007f9113120000)
    libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1
(0x00007f9112ef8000)
    libacl.so.1 => /lib/x86_64-linux-gnu/libacl.so.1 (0x00007f9112cf0000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f9112920000)
    libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3
(0x00007f91126b0000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f91124a8000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f9113658000)
    libattr.so.1 => /lib/x86_64-linux-gnu/libattr.so.1
(0x00007f91122a0000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
(0x00007f9112080000)
```

O linux, através de um programa interno chamado `ld.so` , consegue mapear o caminho das bibliotecas que a aplicação necessita. O

`ld.so` procura basicamente as bibliotecas em dois diretórios que são:

```
/lib
/usr/lib
```

Dentro do diretório `/etc` temos dois arquivos importantes:

```
/etc/ld.so.cache
/etc/ld.so.conf
```

O `ld.so.cache` é o arquivo que mapeia todas as bibliotecas conhecidas pelo linux e faz o vínculo com a aplicação. O `ld.so.conf` é o arquivo principal de configuração, que aponta para o diretório `ld.so.conf.d` .

Para recompilar o cache do `ld.so` , usamos o comando:

```
ldconfig
```

Com a opção `-p` podemos visualizar todas as bibliotecas que estão reconhecidas no linux no momento:

```
ldconfig -p | less
```

Incluir uma biblioteca

Para fim de exemplo, vou criar um diretório em `/tmp` chamado `lib` :

```
mkdir /tmp/lib
```

E agora vou apontar este caminho para o meu `ld.so` , abra o arquivo:

```
vim /etc/ld.so.conf
```

E adicione a linha:

```
/tmp/lib
```

Salve o arquivo e feche-o. Agora execute:

```
ldconfig
```

Uma outra maneira **temporária** de se configurar um caminho de bibliotecas é usando a variável `LD_LIBRARY_PATH` , no entanto, só funciona no seu Shell atual. Ou seja, onde você está digitando os comandos. Caso abra um novo Shell, isso não funcionará.

```
export LD_LIBRARY_PATH=/tmp/lib/
```

Conclusão

A certificação cobra apenas uma pergunta sobre biblioteca no exame sendo este o tópico de menor peso na prova. Sobre o gestor de boot atente-se aos locais onde estão localizados os arquivos de configuração.

Capítulo 13
Gerenciamento de pacotes

A certificação trabalha com o padrão debian e o padrão RPM/Red Hat. Os comandos base para a certificação são o `dpkg`, `apt-get`, `rpm` e `yum`.

Algumas considerações importantes:

1. Cada sistema deve usar um sistema de gerenciamento único.
2. Um pacote é um conjunto de arquivos relacionados a uma aplicação; executáveis, configurações, bibliotecas, manuais etc., além de informações sobre dependências.
3. Os gerenciadores de pacotes mantêm uma base de dados relacionando todos os pacotes, seus arquivos e dependências.

O comando dpkg

Você pode visualizar que o `dpkg` tem uma série de opções com o comando:

```
dpkg --help
```

Para ver a lista de programas instalados, use:

```
dpkg -l
```

Para um pacote específico:

```
dpkg -l bash
```

Outra opção que lista os pacotes instalados é:

```
dpkg --get-selections
dpkg --get-selections | grep bash
```

Para exemplificar, vou realizar a instalação do Shell KSH via `dpkg`. Primeiro realizei o download (verifique sua versão de arquitetura):

```
wget http://ftp.us.debian.org/debian/pool/main/k/ksh/ksh_93u+20120801-1_i386.deb
```

Para encontrar este link, eu pesquisei no google: "Download ksh debian package". Antes mesmo de realizar a instalação deste pacote é possível visualizar os arquivos que o contêm, para isso, utilize:

```
dpkg -I ksh_93u+20120801-1_i386.deb
```

Uma outra opção seria:

```
dpkg --contents ksh_93u+20120801-1_i386.deb
```

Para instalar um pacote no debian, usamos:

```
dpkg -i ksh_93u+20120801-1_i386.deb
```

Para visualizar o que foi instalado (status do pacote), usamos:

```
dpkg -l ksh
```

Para ver a lista de arquivos usados por esta aplicação, usamos a opção em maiúscula:

```
dpkg -L ksh
```

Para saber a origem de um arquivo, ou seja, a qual pacote ele pertence, podemos usar:

```
dpkg -S /usr/share/menu/ksh
```

Para remover o pacote, usamos:

```
dpkg -r ksh
```

Se você rodar:

```
dpkg -l ksh
```

Vai observar que ele ainda mantém informações do pacote na base de dados. Porém, ao executar:

```
dpkg --get-selections |grep ksh
```

Você vai observar que o pacote não está instalado. Para remover o pacote e qualquer informação da base de dados, usamos:

```
dpkg --purge ksh
dpkg -P ksh
```

Observe que se rodar:

```
dpkg -l ksh
```

Já não vai encontrar nenhuma informação na base de dados.

13.1 Comandos do apt

O `apt` utiliza um repositório, com os endereços dos locais onde os pacotes serão encontrados. Esse repositório fica localizado em:

```
/etc/apt/sources.list
```

O primeiro comando que vamos observar é o `apt-cache`:

```
apt-cache
```

Observe que ele mostra algumas funções que podemos usar, por exemplo, a opção:

```
apt-cache pkgnames
```

Essa opção mostra uma lista de pacotes instalados no sistema. Para ver detalhes de um pacote em especial, usamos:

```
apt-cache show vim
```

Para ver, por exemplo, de quais pacotes (dependências) o VIM precisa para funcionar, usamos:

```
apt-cache depends vim
```

O comando apt-get

O primeiro comando que devemos observar do `apt` é o `apt-get update` .

```
apt-get update
```

Sua função é ler o arquivo de repositórios (`/etc/apt/sources.list`) e atualizar a base de dados, com as informações mais recentes de atualizações dos repositórios. Assim, sempre que você solicitar a instalação de um pacote, ele saberá exatamente onde procurar e qual a última versão disponível para a instalação.

Observação: o comando `apt-get update` não atualiza o sistema operacional, ele somente atualiza a lista de programas disponíveis nos repositórios.

Atualizando o sistema operacional

Para atualizar o sistema operacional, usamos o comando:

```
apt-get upgrade
```

Esse comando verificará e atualizará todos os pacotes disponíveis no seu sistema, buscando as versões mais atualizadas no repositório. Outro método é usando o comando:

```
apt-get dist-upgrade
```

A diferença entre os dois é que este atualiza a versão de sua distribuição caso uma nova versão esteja disponível nos repositórios.

Buscando um pacote

Digamos que você precisa de um editor, mas não sabe o nome de nenhum para realizar sua instalação. Uma maneira de procurar por todos os editores disponíveis no repositório é usando o comando:

```
apt-cache search editor
```

Nesse caso, você terá uma lista com todos os pacotes que tenha a string "editor". Quando encontrar um que seja adequado, instale-o usando o comando:

```
apt-get install nome_do_pacote
```

Instalando e removendo um pacote

Para a instalação de um pacote, utilize o comando:

```
apt-get install nome_do_pacote
```

Instalar mais de um pacote de uma vez também é possível, basta dar um espaço em cada um deles:

```
apt-get install nome_do_pacote nome_do_pacote
```

Para remover o pacote, utilize o comando:

```
apt-get remove nome_do_pacote
```

Para remover o pacote, removendo também informações da base de dados e arquivos de configurações, utilize:

```
apt-get purge nome_do_pacote
```

Fazer apenas o download do pacote

Uma opção que poucos conhecem é a possibilidade de se fazer apenas o download do pacote `.deb` , mas não realizar sua instalação. Isso é útil quando você deseja, por exemplo, copiar este instalador para um pendrive e levar a alguma máquina que está sem acesso à internet. Para isso, você pode usar o comando:

```
apt-get --download-only install nome_do_pacote
```

O download do pacote é realizado dentro do diretório `/var/cache/apt/archives` . Toda vez que você instala um pacote, o arquivo de download fica nesse diretório. Para limpar este cache, você pode usar o comando:

```
apt-get clean
```

Comando dpkg-reconfigure

Este comando é utilizado para alguns pacotes que permitem interação com o usuário, como timezone, definição de programas padrão, modelo de teclado, entre outros. Este comando permite refazer a configuração deste pacote, por exemplo:

```
dpkg-reconfigure tzdata
```

Esse comando permite fazer a reconfiguração da timezone. Outro exemplo:

```
dpkg-reconfigure keyboard-configuration
```

Comando dselect

O comando `dselect` não é padrão na maioria das distros. Ele é uma interface para o comando `apt` , através de menus interativos.

```
dselect
```

Comando aptitude

O `aptitude` também tem uma interface de menu, mais complexa do que a do `dselect` :

```
aptitude
```

Os comandos do `aptitude` são parecidos com o do `apt-get` e servem para atualizar os repositórios. Repositórios são os servidores na internet que armazenam as aplicações que vamos instalar em nossa distribuição de Linux.

```
aptitude update
```

Atualizar distro:

```
aptitude safe-upgrade
```

Pesquisar pacote:

```
aptitude search csh
```

Instalar um pacote:

```
aptitude install csh
```

Remover um pacote:

```
aptitde remove pacote
aptitude purge pacote
```

13.2 O aplicativo alien

O alien é um aplicativo que permite a conversão de pacotes `.deb` para `.rpm`, entre outros formatos. Vale lembrar que não é recomendável a utilização de pacotes que não são nativos da sua distribuição de linux.

Para converter um pacote `.deb` para o formato `.rpm`:

```
alien -r nome_do_pacote.deb
```

Para converter um pacote `.rpm` para `.deb`:

```
alien pacote.rpm
```

O padrão do alien é converter pacotes `.rpm` para `.deb`.

13.3 Gerenciador de pacote RPM

Para verificarmos todos os pacotes instalados na máquina, usamos:

```
rpm -qa
```

Para obter informações sobre se um pacote em específico está instalado, você pode usar:

```
rpm -qa bash
```

Para detalhes dessa instalação, o comando seria:

```
rpm -qi bash
```

Para verificar os arquivos que pertencem à instalação, utilize o comando:

```
rpm -ql bash
```

Para visualizar apenas os arquivos de configuração:

```
rpm -qc bash
```

E os arquivos de documentação:

```
rpm -qd bash
```

Para você verificar um arquivo e saber quem é o pacote dono deste arquivo, utilize:

```
rpm -qf /etc/skel/.bashrc
```

Para os próximos exemplos, vamos necessitar de um pacote `.rpm`. Para isso, podemos usar comando `yum` que é semelhante ao `apt-get` com a opção de apenas fazer o download de um pacote sem instalá-lo:

```
yum install --downloadonly --downloaddir=/tmp/programa zsh
```

Nesse exemplo, estou fazendo download do pacote `zsh` dentro do diretório `/tmp/programa`.

Agora que temos o pacote `zsh.... rpm` dentro do diretório `/tmp/programa`, vamos visualizar suas informações:

```
rpm -qlp zsh...rpm
```

Podemos também ver mais detalhes:

```
rpm -qip zsh...rpm
```

Para instalar um pacote, usamos:

```
rpm -i zsh...rpm
```

Caso você já tenha um pacote instalado e deseja instalar uma nova versão que você baixou, use a opção:

```
rpm -U nome_do_pacote.rpm
```

Uma opção interessante que pode ser usada durante a instalação de um pacote é a exibição de uma barra de progresso:

```
rpm -ivh nome_do_pacote.rpm
```

O comando `rpm` não resolve dependências de pacotes. Quando um pacote possui dependências, ele precisa de outra aplicação e/ou biblioteca de sistema para funcionar que não se encontra no seu sistema. O `rpm` não instala este pacote.

Existe uma forma de você instalar um pacote ignorando as dependências, mas isso é arriscado porque sua aplicação pode não funcionar corretamente.

```
rpm -ivh --nodeps nome_do_pacote.rpm
```

Além das opções que já observamos sobre a instalação de um pacote, podemos "forçar" uma instalação. Nesse caso, o sistema vai ignorar qualquer erro que possa estar ocorrendo ao tentar instalar este pacote do modo normal. Vale ressaltar que instalar a aplicação desta forma não garante que ela funcionará corretamente.

```
rpm -ivh --force nome_do_pacote.rpm
```

Antes de instalarmos um pacote, é possível fazer um teste para saber se ele será instalado corretamente em seu sistema, para esse teste usamos o comando:

```
rpm -ivh --teste nome_do_pacote.rpm
```

Agora que já sabemos todas estas formas de se instalar um pacote, vamos aprender como podemos removê-lo. Para isso, use:

```
rpm -e zsh
```

Assim como realizamos na instalação com barra de progresso, podemos remover um pacote, visualizando uma barra de progresso, neste caso o comando é:

```
rpm -evh zsh
```

13.4 Gerenciador de pacotes yum

Para instalar um pacote, use o comando `yum install`, como em:

```
yum install mysql-server
```

Para removê-lo posteriormente, use:

```
yum remove mysql-server
```

O `yum` possui também um recurso de busca que é bastante útil quando você está procurando por um pacote, mas não sabe o nome exato, ou em casos de pacotes que possuem nomes diferentes em relação a outras distribuições. Use o comando `yum search`, seguido por alguma palavra ou expressão, que faça parte do nome do pacote ou descrição, como em:

```
yum search samba
```

Esse comando retorna um relatório contendo todos os pacotes relacionados, incluindo o texto de descrição de cada um. Isso resulta geralmente em uma lista relativamente longa.

Para fazer uma busca mais restrita, procurando apenas nos nomes dos pacotes, use o parâmetro `list`, como em:

```
yum list httpd
```

Ele é bem menos informativo, retornando apenas os pacotes que possuem `httpd` no nome, sem pesquisar nas descrições.

Uma terceira opção é o `provides`, que mostra pacotes que incluem um determinado arquivo, pesquisando não no nome ou na descrição, mas sim no conteúdo dos pacotes. Ele é bastante útil em casos em que você precisa de alguma ferramenta ou biblioteca que faz parte de outro pacote maior, como em:

```
yum provides mcedit
```

Em caso de dúvida, você pode verificar se um determinado pacote está instalado e qual é a versão usando o comando `rpm -q`, como em:

```
rpm -q samba
samba-3.0.25b-0.el5.4
```

Para atualizar um pacote já instalado, use o comando `yum update`, como em:

```
yum update samba
```

O comando `yum install` também pode ser usado para atualizar pacotes. A diferença entre o `install` e o `update` é que o `update` se limita a atualizar pacotes já instalados. Ao perceber que o pacote solicitado não está instalado, ele exibe um aviso e aborta a instalação, como no exemplo a seguir. Isso reduz a possibilidade de você acabar instalando um novo serviço por engano:

```
yum update mysql-server
Loading "installonlyn" plugin
Setting up Update Process
Setting up repositories
Reading repository metadata in from local files
Could not find update match for mysql-server
```

Outra observação é que, depois de atualizar um serviço, é necessário recarregar o serviço (como em `service smb restart`) para que a nova versão passe a ser usada. Esta é mais uma pequena

diferença com relação às distribuições derivadas do debian, onde os serviços são reiniciados de forma automática depois de atualizados.

Para atualizar todo o sistema, comece usando o parâmetro `check-update`, que lista as atualizações disponíveis:

```
yum check-update
```

Se usado sem especificar um pacote, o `update` vai atualizar de uma vez só todos os pacotes do sistema, de forma similar ao `apt-get upgrade` do debian:

```
yum update
```

Existe ainda o comando `yum upgrade`, que é um pouco mais incisivo, incluindo também pacotes marcados como obsoletos (que não existem mais na versão atual). Ele é útil em casos em que é necessário atualizar uma versão antiga do sistema:

```
yum upgrade
```

É possível também fazer com que o `yum` atualize o sistema automaticamente todas as madrugadas. Para isso, basta ativar o serviço `yum` e configurá-lo para ser ativado durante o boot:

```
chkconfig yum on
service yum start
```

Isso faz com que a atualização seja agendada através do `cron` e seja (por padrão) executada todos os dias às 4:02 da manhã, como especificado no arquivo `/etc/crontab`.

Conclusão

Esse tópico tem um grande peso no exame, em média sete perguntas são sobre gerenciamento de pacotes. Isso faz sentido, já que no dia a dia de um administrador instalar e remover programas é tarefa trivial. Recomendo que você treine por diversas vezes os comandos aqui apresentados, pois será um diferencial para uma boa pontuação na prova.

Capítulo 14
Dispositivos, sistemas de arquivos linux e FHS

A certificação utiliza o utilitário `fdisk`, a principal ferramenta de particionamento de discos. Para visualizar os dispositivos e partições, como já mencionamos no livro, podemos o usar o comando:

```
fdisk -l
```

Quando criamos uma partição, devemos formatá-la com um sistema de arquivos. No caso do linux, o comando é:

```
mkfs.ext4 /dev/sdb1
```

O utilitário `mkfs` é o responsável por aplicar o sistema de arquivos, ou seja, formatar o seu disco, de forma que você possa utilizar essa partição. Ao digitar `mkfs` e usar a tecla `< tab >`, você verá todas as opções disponíveis em seu sistema. Você pode formatar discos em NTFS, FAT32, Ext2, 3, 4 entre outros formatos.

Mantendo a integridade do sistema de arquivo

Outra maneira de visualizarmos nossas partições é com o comando `df`.

```
df
```

Uma opção melhor para visualizar essas informações é utilizando o parâmetro `-h`:

```
df -h
```

Essa opção é útil porque mostra o tamanho em Mb, Gb e Tb. Para visualizar quais são os sistemas de arquivos em suas partições, utilize:

```
df -hT
```

Eu também posso visualizar as partições que eu tenho montadas com um tipo de sistema de arquivos:

```
df -ht ext4
```

É possível também verificar o número de INODE. Mas o que é um INODE?

Creio que você já deve ter se perguntado por que o pendrive de 1.5G mostra disponível apenas 1.38G, ou ainda, por que o seu recém-adquirido HD de 80G disponibiliza um pouco menos, por volta de 78G. Afinal, para onde vão esses bytes que na hora de venda são anunciados, mas que, quando você vai usar, não estão disponíveis?

Calma, esses bytes não sumiram e muito menos você foi enganado por uma propaganda enganosa. Esse espaço "invisível" é reservado para um tipo especial de estrutura de dados chamado *INODE*.

Essa estrutura é responsável por conter informações básicas sobre seus arquivos e pastas, como permissões de acesso, identificação dos donos dos arquivos, data e hora do último acesso e alterações, tamanho e, mais importante, os famosos ponteiros para o arquivo em si. De modo geral, o INODE é a identidade de um arquivo ou diretório, é uma identificação única para ele.

Ao ler qualquer arquivo ou diretório, o kernel trata de ler primeiramente o INODE do arquivo para depois realizar a leitura do arquivo. Ao ler essa estrutura, são checadas as permissões e, em caso acesso negado, a famosa mensagem *"Permission denied"* é retornada. Caso contrário, o procedimento segue normalmente.

Listando arquivos e verificando o seu INODE:

```
ls -li
```

Pode acontecer de você ainda ter um pequeno espaço no disco, mas não conseguir mais gravar informações no disco, por falta de INODE. Para verificar, execute:

```
df -i
```

O comando `du` mostra quanto espaço está sendo utilizado por cada diretório.

```
du
```

É possível listar também, além dos diretórios, os arquivos:

```
du -a
```

Uma forma melhor é usar:

```
du -ah
```

A forma mais simples de se verificar o tamanho dos diretórios é usando:

```
du -hs *
```

Ou especificando um diretório:

```
du -hs /home
```

Para limitar a quantidade de subdiretórios em que o comando vai se aprofundar, podemos usar:

```
du -h --max-depth=2
```

Nesse exemplo, o comando vai verificar até dois níveis de diretório.

14.1 Comandos de análise de sistema de arquivo

Um comando bem conhecido é o `fsck - File System Check` . Ele faz correções de blocos com defeito, que normalmente acontecem quando a energia é cortada de forma abrupta.

Observação: o `fsck` só funciona quando a partição não está montada, ou se ela estiver como somente leitura.

O `fsck` tem variações para outros sistemas de arquivos, além do ext4 padrão do linux. Você pode verificar as disponibilidades usando o comando:

```
ls -l /sbin/fsck*
```

O comando para realizar a análise em um sistema ext4 é:

```
fsck.ext4 /dev/sdb2
```

Comando dumpe2fs

Para obter informações sobre uma determinada partição, usamos o comando `dumpe2fs` . Por exemplo:

```
dumpe2fs /dev/sda1 -h
```

Com esse comando, podemos visualizar diversas informações sobre a partição. Uma informação interessante é "Reserved Block Count" que mostra quanto espaço da partição está reservado para não ser usado (o padrão em média é 5%).

O comando `dumpe2fs` só funciona para os sistemas ext2, 3 e 4.

Partições xfs

Para partições `xfs` , podemos obter informações usando:

```
xfc_info /dev/sc1
```

Para gerar um `dump` com informações da partição e gerar um arquivo, podemos usar:

```
xfs_metadump /dev/sdc1 /tmp/arquivodump_xfs
```

Setar informações da partição

Para setar as informações que obtemos no `dumpe2fs` , usamos o comando `tune2fs` . Assim como o `dumpe2fs` , o `tune2fs` só funciona em partições ext2, 3 e 4.

Para colocar um label em uma partição, executamos:

```
tune2fs -L "Partição de backup" /dev/sdb1
```

Você pode verificar o resultado executando o comando:

```
dumpe2fs /dev/sdb1
```

Para definir o tempo da checagem do sistema de arquivos, usamos:

```
tune2fs -i 1m /dev/sda1
```

Nesse exemplo, defini que a checagem ocorrerá a cada 1 mês.

Os sistemas de arquivos modernos possuem uma implementação chamada *journaling*. Um sistema de arquivos com *journaling* (em português, diário) é aquele que rastreia as alterações ainda não confirmadas na parte principal do sistema de arquivos, registrando as intenções de tais alterações em uma estrutura de dados conhecida como *log de dados* (diário), que geralmente é um log circular.

No caso de uma falha do sistema ou falha de energia, esses sistemas de arquivos podem ser colocados on-line novamente mais rapidamente com uma probabilidade menor de serem corrompidos. Sistemas anteriores ao ext4, como Ext3 e Ext2 não possuem *journaling*. No entanto, é possível converter esses sistemas usando:

```
tune2fs -j /dev/sdb4
```

14.2 Controle de montagem e desmontagem de partições

Agora que já criamos uma partição e aplicamos um sistema de arquivos, é necessário criar um ponto de montagem para acessar os arquivos dessa partição. Um ponto de montagem é o diretório que eu vou associar à partição.

Podemos visualizar as partições que estão montadas usando:

```
df -h
```

Ou através do comando:

```
mount
```

Para montarmos uma partição de forma temporária (perde no reboot da máquina), usamos os comandos `mount` / `umount` . Para que a partição esteja disponível após o reboot, devemos editar o arquivo `/etc/fstab` .

Para montarmos todas as partições que estão no arquivo `/etc/fstab` usamos o comando:

```
mount -a
```

Para desmontar todas as partições que estejam no arquivo `/etc/fstab` , mas que não estão em uso, usamos:

```
umount -a
```

Criando e montando a partição

No meu sistema, eu possuo a partição `/dev/sdb1` com o sistema de arquivos ext4. O primeiro passo é criar um diretório que será usado como ponto de montagem. Para seguir um padrão do FHS, vou criar este ponto de montagem dentro do diretório `/mnt` (mount).

```
mkdir /mnt/sdb1
```

Observe que criei o diretório com o mesmo nome da partição que vou associar (montar) somente para facilitar, pois poderia criar esse

diretório com qualquer nome. Aliás, é possível montar em qualquer diretório, só criei em `/mnt` para seguir um padrão.

Agora que já tenho o diretório, vou executar o comando para montar a partição:

```
mount -t ext4 /dev/sdb1 /mnt/sdb1
```

Para desmontar essa partição, podemos usar:

```
umount /dev/sdb1
```

Ou

```
umount /mnt/sdb1
```

Fstab

Abra o arquivo `fstab` com o seu editor de texto de preferência:

```
vim /etc/fstab
```

Vemos a seguir um exemplo de uma linha do arquivo `Fstab` :

```
/dev/cdrom   /media/cdrom   auto ro,noauto,user,exec   0   0
```

A primeira coluna, `/dev/cdrom` , corresponde ao hardware que queremos montar. A segunda coluna significa o lugar onde queremos montar o hardware (lugar em que aparecerá o seu conteúdo), no caso, o cdrom `/media/cdrom` . Na terceira coluna, `auto` , começa a complicar um pouco...

auto

Essa opção se refere ao sistema de arquivos da unidade. Se ela é ext3, ext4, ntfs etc. É recomendável deixar a opção `auto` , pois o sistema de arquivos será detectado automaticamente. Em seguida, teremos a quarta coluna `ro,noauto,user,exec` . Ela é composta de valores em sequência e sem espaço entre elas. Vamos tentar entender o que significam esses valores.

ro e rw

O `ro` significa *read-only*, ou somente leitura, e `rw` significa *read-write*, ou leitura e escrita.

auto e noauto

Com a opção automática (`auto`), o dispositivo será montado automaticamente (na inicialização). `auto` é a opção padrão. Se você não quer que o dispositivo seja montado automaticamente, use a opção `noauto` . Com `noauto` , o dispositivo pode ser montado apenas manualmente.

user e nouser

Estas são opções muito úteis. A opção do usuário (`user`) permite que os usuários normais montem o dispositivo, ao passo que `nouser` permite apenas ao root montar o dispositivo. `nouser` é o padrão, que é uma das principais causas de dor de cabeça para os novos usuários linux. Se você não é capaz de montar o seu cdrom, disquete, partição do Windows, ou algo como um usuário normal, adicione a opção do usuário no `/etc/fstab` .

exec e noexec

`exec` permite executar binários que estão nessa partição, enquanto `noexec` não permite que você faça isso.

`noexec` pode ser útil para uma partição que contém os binários que você não deseja executar no seu sistema, ou que nem sequer pode ser executado em seu sistema. Este poderia ser o caso de uma partição Windows. `exec` é a opção padrão, que é uma coisa boa. Imagine o que aconteceria se você acidentalmente utilizasse a opção `noexec` com sua partição raiz do Linux.

14.3 sync e async

São formas como a entrada e saída para o sistema de arquivos devem ser feitas. `sync` significa que é feito de forma síncrona. Se você olhar para o `fstab` da unidade de disquete, você vai perceber que esta é a opção utilizada. Significa que quando você, por exemplo, copiar um arquivo para o disquete, as alterações são gravadas no disquete, no mesmo tempo que você emitir o comando de cópia.

No entanto, se você tem a opção `async` em `/etc/fstab`, entrada e saída são feitas de forma assíncrona. Agora, quando você copiar um arquivo para o disquete, as alterações podem ser fisicamente gravadas muito tempo depois de emitir o comando.

Isso não é ruim, e pode ser favorável, mas pode causar alguns acidentes desagradáveis: se você acabou de remover o disquete sem desmontar primeiro, o arquivo copiado pode não existir fisicamente no disquete ainda! `async` é o padrão. No entanto, use a sincronização com o disquete, especialmente se você está acostumado ao modo como ele é feito no Windows e tem uma tendência a retirar os disquetes antes de desmontar primeiro.

Você pode, ainda, em vez de usar todas essas variáveis, utilizar uma opção que reúna todas as opções padrão na quarta coluna.

defaults

A opção `defaults` (padrão) já define as configurações `rw`, `exec`, `auto`, `nouser` e `async`.

```
/dev/hda2 / ext4 defaults 1 1
```

Na quinta coluna, temos um número que pode ser 0 ou 1. Se for 0, a unidade não será checada pela análise de sistema de arquivos do linux rotineiramente.

A sexta coluna é apenas a ordem em que será verificado o sistema de arquivos. Se for 1, será o primeiro. Se for 2, será a segunda unidade a ser verificada.

Exemplos de linhas no fstab

Observe um exemplo de algumas linhas do arquivo `/etc/fstab` :

```
/dev/hda2 /    ext4 defaults 1 1
/dev/hdb1 /home ext4 defaults 1 2
UUID=24235192-541f-4a26-ab5b-0b0e00ee93c1 /home/juliano/hd2 ext4
defaults,user,exec 0 2
```

O `UUID` apresentado acima é uma referência para a partição. Você pode obter este número usando o comando:

```
blkid /dev/sdb1
/dev/sda1: UUID="24235192-541f-4a26-ab5b-0b0e00ee93c1" BLOCK_SIZE="4096"
TYPE="ext4" PARTUUID="16185c75-01"
```

14.4 Gerenciando permissões e propriedades do arquivo

Um arquivo no linux pertence ao usuário que o criou e ao grupo dono ou associado a esse arquivo. Um arquivo tem 12 bits com as informações das permissões deste arquivo.

Esses bits são divididos em quatro grupos. Sendo que os primeiros 3 bits são para configurações de permissões especiais, que veremos mais à frente; os próximos 3 bits, para configurações de permissões do usuário; os 3 outros, para configurações do grupo do arquivo; e os bits finais, para configurações das permissões para outros usuários. Cada arquivo no sistema possui três permissões básicas:

Caractere e valor	Descrição
r (4)	Permissão de leitura - Read
w (2)	Permissão para escrita - Write
x (1)	Permissão de execução - Execute

A cada permissão é atribuído um valor, mostrado entre parênteses, que será utilizado para a definição de permissões. Além disso, cada arquivo contém três conjuntos de permissões, sendo eles:

- Permissão do dono (u) - “user” do arquivo;
- Permissão do grupo (g) - “group” ao qual o arquivo pertence;
- Permissão de outros (o) - “others”, aqueles que não pertencem ao grupo e não são os donos do arquivo.

Sendo assim, considere a seguinte saída do comando `ls -l` para um arquivo:

```
-rw -r --r - - 1 root root 0 Jan 15 09:52 arquivo
```

E esta é a saída para um diretório:

```
drwxr -xr - x 2 root root 4096 Jan 15 09:52 diretório
```

Vamos entender o que essas linhas significam:

Caractere	Descrição
-	Indica que é um arquivo comum.
d	Indica que é um diretório.
l	Indica que é um link simbólico.
p	Indica que é um pipe nomeado.
s	Indica que é um socket.
c	Indica que é um dispositivo de caractere.

Caractere	Descrição
b	Indica que é um dispositivo de bloco.

Os próximos três conjuntos de três caracteres indicam as permissões do grupo e depois a permissão de outros.

Exemplo: - rwx(dono) - rwx(grupo) - rwx(outros).

Nesses três conjuntos, se o caractere encontrado for um `-` (hífen), significa que a permissão está ausente, ou seja, não há a respectiva permissão. Se alguma ou todas as letras (r, w e x) forem encontradas, indicará as permissões que o arquivo as tem definidas. Exemplo de permissão ausente:

```
--- --- ---
```

Nenhuma permissão atribuída.

Ou:

```
rwx --- ---
```

Somente o dono tem permissão. Nenhuma permissão atribuída ao grupo e a outros.

Para ampliarmos nosso conhecimento, crie um usuário chamado `aluno` , que terá um grupo padrão também definido como `aluno` . Vamos criar um diretório para testes e criar um arquivo dentro dele. Logue-se como usuário aluno:

```
mkdir /tmp/aula
cd /tmp/aula
touch arquivo
```

Visualize a permissão do arquivo criado:

```
ls -l /tmp/arquivo
-rw -r --r - - 1 aluno aluno 0 2011 -11 -04 12:17 arquivo
```

Sua permissão para o dono é leitura e escrita: `r w -` Sua permissão para grupo é somente leitura: `r - -` Sua permissão para outros é somente leitura: `r--`

Permissão literal

O comando que usamos para alterar os valores das permissões é o `chmod` . Observe:

```
chmod u+rwx /tmp/arquivo
```

Nesse caso, atribuímos permissões de leitura e escrita ao dono do arquivo. Para remover essas permissões, usamos:

```
chmod u-rwx /tmp/arquivo
```

Para adicionarmos permissões ao grupo, usamos:

```
chmod g+rwx /tmp/arquivo
```

E para outros:

```
chmod o+rwx /tmp/arquivos
```

Como podemos observar, o sinal de `+` adiciona, e o sinal de `-` remove as permissões.

O parâmetro `u-rw` é que define o esquema de permissões. A primeira letra indica para qual(is) usuário(s) as permissões estão sendo alteradas. Usamos a letra `u` para indicar o próprio dono, `g` para indicar o grupo, `o` para outros e ainda a letra `a` para indicar todos.

Observe a permissão a seguir:

```
ls -l /tmp/arquivo
-rw- r-- --- 1 aluno aluno 0 2011 -11 -04 12:17 arquivo
```

O dono do arquivo possui permissão de leitura e escrita (`rw`). O grupo possui permissão de leitura (`r`). E outros, não possuem permissões (`---`).

As permissões seguem uma ordem: *dono*, *grupo*, *outros*, ou seja, se você é o *dono*, as permissões que se encaixam no seu perfil é a de dono, mesmo que você pertença ao *grupo*, as permissões de dono prevalecem.

Tente visualizar o arquivo:

```
cat /tmp/arquivo
```

Faça login com um outro usuário e tente ler o arquivo:

```
cat /tmp/arquivo
```

A leitura será possível porque a permissão *outros* possui permissão de leitura. As permissões de grupo permitem que ele abra o arquivo para leitura, mas não para escrita:

```
echo oi >> arquivo
```

Vejamos mais um exemplo. Como usuário aluno, faça:

```
chmod g + w arquivo
```

Esse comando adiciona a permissão de escrita para os usuários que fazem parte do mesmo grupo ao qual o arquivo pertence. As demais permissões não são alteradas.

Vamos testar agora a permissão de execução. Crie um novo arquivo:

```
touch /tmp/diretorio/novo
```

Visualize e acesse o diretório:

```
ls -l /tmp/diretorio
cd /tmp/diretorio
```

Para que os usuários acessem um diretório, ele precisa ter permissão de execução. Vamos remover a permissão de execução de todos os diretórios:

```
chmod a-x /tmp/diretorio
```

Agora tente acessar o diretório:

```
cd /tmp/diretorio
```

Isso não é possível porque a permissão de execução (`x`) foi removida. Sem execução, o diretório não pode ser acessado.

```
ls /tmp/diretorio
```

Visualize as permissões do diretório e do arquivo:

```
ls -ld /tmp/diretorio
drw -r --r - - 2 aluno aluno 4096 2011 -11 -04 17:02 diretorio /

ls -l /tmp/diretorio/novo
-rw -r --r - - 1 aluno aluno 0 2011 -11 -04 17:40 novo
```

Permissão octal

A segunda sintaxe é a forma numérica. Neste caso, o parâmetro que define as permissões é composto de três números de 0 a 7, que correspondem às permissões para o usuário dono, para o grupo e para outros. Cada número é formado pela soma das permissões atribuídas, sendo que execução vale 1, escrita vale 2 e leitura 4. Vejamos um exemplo:

```
chmod 640 /tmp/arquivo
```

Neste caso, estamos atribuindo a permissão de: leitura e escrita 6 (r=4 + w=2) ao usuário dono leitura 4 (r=4) ao grupo 0 (sem permissões) a outros usuários. É importante observar que, quando usamos a forma literal, alteramos apenas o parâmetro especificado, sem mudar as demais permissões. Já na forma numérica, alteramos todas as permissões simultaneamente.

Vamos ver exemplos de permissões.

Comando para atribuir permissão total a um arquivo chamado `naofazer` :

```
touch /tmp/naofazer
chmod 777 /tmp/naofazer
chmod a + rwx /tmp/naofazer
```

Ou

```
chmod u + rwx , g + rwx , o + rwx /tmp/naofazer
```

Verificar a permissão:

```
ls -l /tmp/naofazer
 - rwxrwxrwx 1 root root 0 2011 -11 -06 22:17 naofazer
```

Evite dar permissão total para um arquivo ou diretório a não ser que seja realmente necessário. Não se deve fazer isso em nenhum tipo de arquivo, isso é apenas um exemplo!!!

Comando para retirar a permissão de escrita de todos os usuários do arquivo `naofazer` :

```
chmod 666 /tmp/naofazer
```

Visualize:

```
ls -l /tmp/naofazer
```

Comando para alterar a permissão padrão do arquivo `arquivo` para que todos os usuários possam apenas lê-lo.

```
chmod 444 /tmp/arquivo
```

Visualize:

```
ls -l /tmp/naofazer
```

Utilizando o exemplo anterior do modo literal, vamos alterar a permissão recursivamente, dando permissão total somente para o dono:

Antes:

```
ls -ld /tmp/diretorio
drw -r --r - - 2 aluno aluno 4096 2011 -11 -04 17:02 diretorio /
```

```
ls -l /tmp/diretorio /novo
-rw -r --r - - 1 aluno aluno 0 2011 -11 -04 17:40 novo

# chmod -R 700 /tmp/diretorio
```

Depois:

```
s -ld /tmp/diretorio
drwx ----- - 2 aluno aluno 4096 2011 -11 -04 17:02 diretorio /

ls -l /tmp/diretorio/novo
-rwx ----- - 1 aluno aluno 0 2011 -11 -04 17:40 novo`
```

Umask

O `umask` altera o valor da máscara de criação de arquivos e diretórios. Essa “máscara” é utilizada para definir o “permissionamento” padrão de um arquivo ou diretório quando ele é criado.

Para calcular a `umask` para um diretório, pegue a permissão total a que um diretório pode chegar, 777, e subtraia 022. Esse é o valor padrão de `umask` do sistema.

Para calcular a `umask` para um arquivo, pegue a permissão total que um objeto pode chegar, que é 777, subtraia 111 de 777 (valores de execução do arquivo) e de 0022 (valor padrão de `umask` do sistema no debian).

O valor padrão da `umask` fica armazenado no arquivo `/etc/profile` .

```
umask 0022
```

O primeiro `0` significa modo octal, podemos passar o valor em hexadecimal colocando `0 x` como prefixo.

14.5 Permissões especiais

Há um conjunto especial de permissões conhecido como *bits* especiais. São eles:

O **suid bit** é atribuído a um arquivo binário com permissão de execução, quando desejamos que um usuário qualquer execute o comando com as permissões do usuário dono do comando. Se esse comando pertencer ao usuário `root`, um usuário qualquer vai executá-lo com as permissões de `root` desde que tenha permissões para executá-lo. Por esse motivo, o `suid` constitui uma grande ameaça de segurança e sua utilização deve ser bastante cautelosa.

O **sgid bit** é geralmente atribuível a diretórios. Quando um arquivo é criado dentro de um diretório com `sgid bit` ativado, o conteúdo gravado dentro do diretório herdará o grupo do diretório e não o grupo do usuário que criou tal conteúdo. Este bit especial é muito útil quando utilizamos diretórios para grupos de trabalhos e em servidores de arquivos.

O **sticky bit** era bastante utilizado para realizar otimizações de acesso a conteúdos, entretanto, a partir da série 2.6 do kernel do linux, essa tarefa é realizada diretamente pelo kernel. A única utilidade desse bit, atualmente, é fazer diretórios de utilização comum a todos os usuários, como no `/tmp`. Quando esse bit está ativo em um diretório, todo conteúdo criado dentro dele pertencerá ao criador do conteúdo e, por mais que ele atribua a esse conteúdo permissões totais para todos os usuários, o único que poderá excluir o arquivo ou diretório será o próprio dono, o *root* ou ainda o dono do diretório que tem a permissão.

Para atribuirmos esses bits especiais, procedemos da mesma forma que nas permissões comuns, somando os valores e utilizando o comando `chmod`, mas agora utilizando quatro números, o primeiro número sendo o bit especial, seguido dos três da permissão padrão.

Veja o exemplo:

```
chmod 4000 /tmp/a
chmod 2000 /tmp/b
chmod 1000 /tmp/c

# ls -l /tmp
 ---S ----- - 1 aluno aluno 0 2008 -07 -21 13:50 a
 ------S -- - 1 aluno aluno 0 2008 -07 -21 13:50 b
 ---------T 1 aluno aluno 0 2008 -07 -21 13:50 c
```

O bit especial para o campo de permissões do dono é o `suid`, representado por `s` ou `S`. Para o grupo é `sgid`, também representado por `s` ou `S`. Já o campo de permissões de outros usuários, o `sticky bit`, é representado por `t` ou `T`.

Veja que quando o arquivo ou diretório não tem permissão de execução, o bit especial é representado por uma letra `S` (caixa alta), e quando possuem uma permissão de execução, o bit especial é apresentado como `s` (caixa baixa). O mesmo acontece com o `sticky bit`, mas com a letra `t` e `T`.

Exemplo dos bits especiais com permissão de execução:

```
chmod 4100 /tmp/a
chmod 2010 /tmp/b
chmod 1001 /tmp/c

# ls -l
---s ----- - 1 root root 0 2008 -07 -21 13:50 a
------s -- - 1 root root 0 2008 -07 -21 13:50 b
---------t 1 root root aluno 0 2008 -07 -21 13:50 c
```

Todas as permissões especiais que não contiverem execução são maiúsculas. `S` e `T`.

Exemplos:

suid bit

Para entender a permissão `suid bit`, vamos analisar o comando `passwd`.

```
ls -l /usr/bin/passwd
-rwsr -xr - x 1 root root 31640 2008 -06 -12 20:39 /usr/bin/passwd
```

Observe o `s` na saída do comando. Os nossos usuários comuns só podem mudar sua senha porque o comando `passwd` está com o `suid bit` ativado. Para compreendermos melhor, vamos desabilitar o `suid`:

```
chmod 755 /usr/bin/passwd
ls -l /usr/bin/passwd
```

Tente agora logar-se com algum usuário do seu sistema e trocar sua senha com o comando `passwd`. Você notará que isso não é mais possível, pois o aluno não tem permissão de escrita nos arquivos `/etc/passwd` e `/etc/shadow`.

Com o usuário root, volte à permissão original:

```
chmod 4755 /usr/bin/passwd
ls -l /usr/bin/passwd
```

sgid bit

Crie um diretório com permissão total para qualquer usuário:

```
mkdir /teste
chmod 777 /teste
```

Agora qualquer usuário tem acesso ao diretório `teste`. Como usuário aluno, crie um arquivo no diretório `teste`:

```
touch /teste/numero1
ls -l /teste
-rw -r --r - - 1 aluno aluno 0 2011 -11 -07 11:01 numero1
```

Verifique que o grupo é o mesmo do usuário. Agora com o usuário root altere a permissão do diretório teste, adicionando o `sgid bit`:

```
chmod 2777 / teste
chmod 2777 /teste
```

Novamente como usuário aluno, crie um novo arquivo dentro do diretório teste:

```
touch /teste/numero2
```

Verifique quem é o dono e quem é o grupo do arquivo criado:

```
ls -l /teste
-rw -r --r - - 1 aluno aluno 0 2011 -11 -07 11:01 numero1
-rw -r --r - - 1 aluno root 0 2011 -11 -07 11:19 numero2
```

Visualize que o grupo agora não é mais o do usuário e sim o mesmo do diretório.

stick bit

Com o root, modifique novamente a permissão do diretório teste e adicione o `stick bit` :

```
chmod 1777 /teste
```

Agora modifique as permissões dos arquivos dentro dele dando permissão total para todos os usuários:

```
chmod 777 /teste /*
```

Logue-se como usuário aluno2, visualize as permissões do diretório e dos arquivos:

```
ls -ld /teste
drwxrwsrwt 2 root root 4096 2011 -11 -07 11:19 /teste
```

```
ls -l /teste
 - rwxrwxrwx 1 aluno aluno 0 2011 -11 -07 11:01 /teste/numero1
 - rwxrwxrwx 1 aluno aluno 0 2011 -11 -07 11:19 /teste/numero2
```

Agora que você viu que tem permissão total nos arquivos, tente deletar algum arquivo dentro do diretório teste:

```
rm /teste/numero1
```

Apesar da permissão total no arquivo, não é possível remover devido a permissão do diretório com stick bit.

14.6 Comandos chown, chgrp

Para alterar o nome de um usuário dono de um arquivo, usamos:

```
chown luizinho arquivo
```

Neste caso, o dono do arquivo será `luizinho`. Para alterar o nome do usuário e do grupo, usamos:

```
chown luizinho:financeiro arquivo
```

Nesse exemplo, o grupo foi alterado para `financeiro`. Para trocar somente o grupo, usamos:

```
chown :admin arquivo
```

Troquei de `financeiro` para `admin`, suprimindo o nome do usuário à frente dos `:` (dois pontos).

Para alterar de forma recursiva, usamos:

```
chown -R juliano:financeiro diretorio2
```

Nesse caso, ele alterou o usuário do diretório2 para `juliano` e o grupo para `financeiro`, tanto do `diretorio2` quanto dos subdiretórios.

O comando `chgrp` muda apenas o grupo, exemplo:

```
chgrp -R suporte arquivo2
```

14.7 Criando e alterando links hard e soft - in

O link hard cria um arquivo com o mesmo INODE, sendo assim um mesmo arquivo físico. Crie um arquivo em `/tmp` de nome `teste` e verifique o seu INODE:

```
ls -li teste
```

Agora crie um link hard:

```
ln teste link-hard
```

Verifique que tanto o arquivo `teste` quanto o arquivo `link-hard` possuem o mesmo INODE:

```
ls -li teste link-hard
```

Se você alterar a permissão de um dos dois arquivos, a permissão dos dois é alterada. Link hard nada mais é do que você criar um atalho para o mesmo arquivo físico (mesmo local na partição). Você pode apagar um dos dois e o outro continua. Você apaga apenas uma das referências.

Considerações importantes sobre link hard:

- Só é possível criar link dentro da mesma partição.
- Não é permitido criar hard link para diretórios.

Link simbólico

Link simbólico é um arquivo que aponta para outro arquivo.

```
ln -s /etc/arquivo /opt/link-simbolico
```

No exemplo acima, `/etc/arquivo` agora tem um link simbólico em `/opt/link-simbolico`

Para que você saiba qual arquivo o seu link simbólico aponta, execute:

```
ls -ld /opt/link-simbolico
```

É possível criar um link simbólico para um diretório:

```
ln -s /var /opt/variaveis
```

No exemplo acima, o diretório `/var` agora possui um link simbólico em `/opt/variaveis` .

14.8 Localização de arquivos no linux

O modo mais prático de realizar busca de arquivo ou diretório no linux é através da linha de comando com o comando `find` . Este é o último conteúdo exigido para a prova de certificação LPIC-1 101.

O comando `find` é o mais utilizado para busca de arquivo. Um exemplo:

```
find /home -name juliano
```

Nesse exemplo, ele busca tudo o que tiver o nome `juliano` . Digamos que você tenha arquivos, como `juliano2` e `juliano10` , você pode fazer uma busca recursiva:

```
find /home -name juliano*
```

Para procurar arquivos criados por um determinado usuário, execute:

```
find /home -user aluno
```

Nesse exemplo, o `find` vai procurar todos os arquivos do usuário aluno no diretório `/home` . Podemos buscar arquivos pelo seu tamanho:

```
find /home -size +50M
```

Nesse caso, vai procurar arquivos com mais de 50M.

Comando locate

O comando `locate` busca string através de sua base de dados. Exemplo:

```
locate juliano
```

Nesse exemplo, ele vai buscar tudo o que tenha `juliano` em sua base de dados. Como ele busca na base de dados e não faz varredura do disco, ele é bem mais rápido. Mas é necessário atualizar a base de dados.

```
updatedb
```

O arquivo de configuração do `updatedb` fica em `/etc/updatedb.conf` .

Comando whereis

Este comando localiza executáveis, bibliotecas e páginas de manual. Exemplo:

```
whereis rpm
```

Comando which

O `which` procura por aplicações executáveis que você tenha em seu sistema. Mas, atenção. Ele somente procura nos diretórios que estão no PATH e mostra somente o primeiro retorno que ele encontrar.

```
which rpm
```

Comando type

O comando `type` mostra o tipo de um executável. Exemplos:

```
type ls
type cd
```

Conclusão final

Planeje um dia só para estudar e, de preferência, que não seja um dia antes da prova. Isso vai ajudar você a processar todo o conteúdo, a estudar com tranquilidade e concentração. Quando nos programamos, sobra tempo para os imprevistos.

Então, se necessário, reserve um tempo para revisar alguns pontos sobre os quais você ainda não tem total segurança.

Em meu site www.profjulianoramos.com.br, disponibilizo um curso gratuito em videoaulas para a prova 101. Esse curso usa este livro como base.

Capítulo 15
Agradecimentos

Agradeço a você leitor ou leitora, desejando-lhe muita sorte no exame e na vida. Sinta-se livre para me adicionar nas redes sociais e buscar qualquer orientação que estiver ao meu alcance.

Escrever um livro não é uma tarefa fácil, mas, graças a Deus, tenho pessoas incríveis ao meu lado. Agradeço à Casa do Código, que aproxima o autor do leitores e leitoras disponibilizando um ambiente incrível para escrever, revisar e distribuir conteúdo.

Agradeço ao Professor Cicero Woshington, revisor técnico deste livro e um grande amigo e colega de profissão, agradeço sempre à minha filha, que demonstrou paciência enquanto eu realizava este trabalho. Giovanna, você é minha fonte de inspiração e superação. Agradeço a minha irmã Keila Ramos, que está sempre ao meu lado nos momentos difíceis.

Mas o mais importante, agradeço a Deus por me conceder um ano de superação, em meio a tempestade esteve no banco comigo.

E, como sempre digo em meus vídeos no YouTube e em minhas aulas, ***"A vida é boa e sempre vai dar certo"***.

Prof. Juliano Ramos.

Você pode me encontrar em:

https://youtube.com/profjulianoramos

https://www.linkedin.com/in/profjulianoramos/

http://www.profjulianoramos.com.br